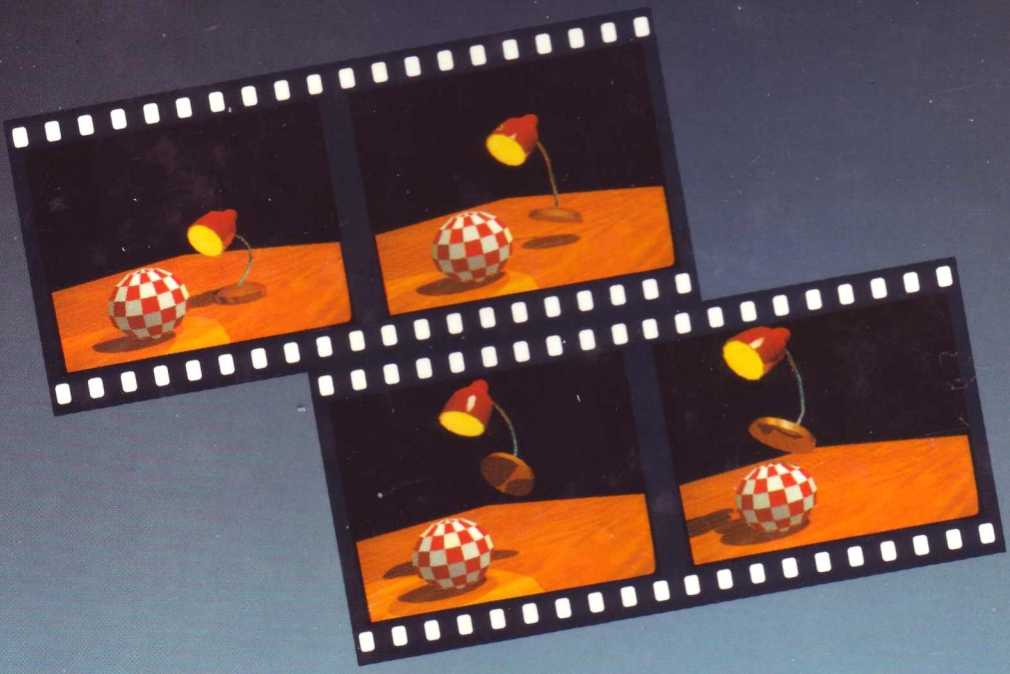
Markt&Technik

Carsten Fuchs

BOOK-WARE

AMIGA

REFLECTIONS ANIMATOR



Fotorealistische Animationen im Raytracing-Verfahren

Auf Diskette enthalten:
Reflections Animator – der einfache Weg, anspruchsvolle Filmsequenzen auf dem Amiga zu erstellen. Ergänzt durch wertvolle Beispieldateien und Hilfsprogramme.

Carsten Fuchs

AMIGA

REFLECTIONS ANIMATOR

Fotorealistische Animationen
im Raytracing-Verfahren

Markt&Technik Verlag AG

CIP-Titelaufnahme der Deutschen Bibliothek

Fuchs, Carsten:
Reflections animator : fotorealistische Animationen im Raytracing-Verfahren / Carsten Fuchs. –
Haar bei München : Markt-und-Technik-Verl., 1991
(Bookware)
ISBN 3-89090-234-0

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten ist nicht zulässig.

Commodore AMIGA ist eine Produktbezeichnung der Commodore Büromaschinen GmbH, Frankfurt,
der ebenso wie der Name Commodore Schutzrecht genießt.
Der Gebrauch bzw. die Verwendung bedarf der Erlaubnis der Schutzrechtsinhaberin.
AMIGA ist ein eingetragenes Warenzeichen der Commodore-AMIGA Inc., USA
Sculpt 3-D ist ein eingetragenes Warenzeichen der Byte by Byte Corporation, USA
Silver ist ein eingetragenes Warenzeichen der Impulse Inc., USA

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

94 93 92 91

ISBN 3-89090-234-0

© 1991 by Markt&Technik Verlag Aktiengesellschaft,
Hans-Pinsel-Straße 2, D-8013 Haar bei München/Germany
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Dieses Produkt wurde mit Desktop-Publishing-Programmen erstellt
und auf der Linotronic 300 belichtet
Druck: Triltsch, Würzburg
Printed in Germany

Inhaltsverzeichnis

	Vorwort	**9**
1	**Überblick**	**11**
1.1	Einige wichtige Begriffe	13
1.1.1	Koordinatensysteme	13
1.1.2	Baum	13
1.1.3	Skelett	14
1.1.4	Gelenke	14
2	**Einführung**	**15**
2.1	Kopieren von »Reflections-Animator«	15
2.2	Installieren der neuen Programme	15
2.2.1	Installieren auf Festplatte	15
2.2.2	Installieren auf zwei Disketten	15
3	**Animator**	**17**
3.1	Überblick	17
3.2	Die Fenster des »Animator«	18
3.2.1	Das Plotfenster	18
3.2.2	Das Statusfenster	18
3.2.3	Das Sequenzfenster	18
3.3	Kurzübersicht Menüs	19
3.3.1	Menüpunkt »Projekt«	19
3.3.2	Menüpunkt »Plotten«	20
3.3.3	Menüpunkt »Körper«	21
3.3.4	Menüpunkt »Special«	21
3.3.5	Menüpunkt »Sequenz«	21
3.3.6	Menüpunkt »PreView«	23
3.3.7	Menüpunkt »Skelett«	23
3.4	Koordinatensystem und Bäume	24
3.4.1	Koordinatensysteme	24

3.4.2	Erzeugung des Körperbaums	27
3.5	Einlesen von »Construct«-Szenen	28
3.6	Die Darstellungsarten (»Plotten«-Menü)	33
3.6.1	Die Funktion Plotkörper (k)	33
3.6.2	Die »Hidd«-Funktionen	33
3.6.3	Gitter	33
3.6.4	Wire-Frame (w)	34
3.6.5	Sonstige	34
3.6.6	Zoom-Gadget	34
3.7	Das »Körper«-Menü	36
3.7.1	Ausrichten (a)	36
3.7.2	Eigenschaften	36
3.7.3	clear_eigenschaften	36
3.7.4	Körpernamen (n)	37
3.7.5	Gelenknamen	37
3.7.6	Zentrieren (z)	37
3.7.7	Lichtfarbe	37
3.8	Veränderungen an Körpern	37
3.8.1	Der Arbeitskörper »Work«	37
3.8.2	Verändern aus der Perspektivsicht	40
3.9	Eine kurze Einführung in Sequenzen	41
3.10	Erzeugung einfacher Sequenzen	43
3.11	Das Sequenz-Fenster	45
3.11.1	Die Gadgets im Sequenzfenster	46
3.12	Das Sequenzmenü	49
3.13	Erzeugen von Testanimationen	55
3.14	Skelette	57
3.15	Gelenke	60
3.15.1	Gelenke im Sequenz-Menü	66
3.15.2	Gelenke in Gadgets	68
3.15.3	Die Eigenschaften im Körper-Menü	69
3.15.4	Take_Snaps	70
3.15.5	Gelenkfunktionen im Special-Menü	71
3.16	Lichtsequenzen	72
3.17	Das Projekt-Menü	73
3.18	Tips zur Handhabung der Kamera	74
3.18.1	Feststehende Kamera	75
3.18.2	Bewegte Kamera	75
3.19	Körperbaum und Sequenzen	76
3.20	Sequenzen: Fragen und Antworten	79

3.21	Beachtenswertes zu Gelenken	81
3.22	Verzerrungen und Verbiegungen	83
3.23	Tips und Tricks mit Sequenzen	85
3.23.1	Plötzliches Auftauchen oder Verschwinden	85
3.23.2	Auftauchen mit anderen Körpern	87
3.23.3	Sprünge in weichen Kurven	88
4	**Make_Anim – Erzeugen von Animationen**	**91**
4.1	Einleitung	91
4.2	Der Animationsauftrag (Job)	92
4.3	Datensicherheit und Abbrechen der Animation	98
4.3.1	Abbrechen oder Unterbrechen einer Animationsberechnung	99
4.4	Kombination von »Render«, »Show« und »Pack«	100
4.4.1	»Render«, »Show« und »Pack« eingeschaltet	101
4.4.2	»Render« eingeschaltet, »Show« und »Pack« ausgeschaltet	101
4.4.3	»Render« und »Show« eingeschaltet, »Pack« ausgeschaltet	101
4.4.4	»Show« eingeschaltet, »Render« und »Pack« ausgeschaltet	102
4.4.5	»Pack« eingeschaltet, »Render« und »Show« ausgeschaltet	102
4.4.6	»Show« und »Pack« eingeschaltet, »Render« ausgeschaltet	103
4.4.7	Hinweise zum Speicherverbrauch	103
4.4.8	Weitere Hinweise	103
5	**Der Packer**	**105**
5.1	Grundlagen	105
5.2	Nachbearbeitung eines Bildes	106
5.3	Aufruf und Handhabung	109
5.4	Abschätzen des Speicherverbrauchs	111
6	**Scan**	**115**
6.1	Einleitung	115
6.2	Unterschiede zu »Beams«	115
6.3	Aufruf von »Scan«	116
7	**Play**	**119**
7.1	Einleitung	119
7.2	Aufruf von »Play«	120

8	**Der Manager**	**123**
8.1	Einleitung	123
8.2	Erweiterungen zum »Reflections-Manager«	123
9	**Utilities**	**125**
9.1	merge_anim	125
9.2	split_anim	126
9.3	join_anim	127
9.4	anim_time	127
10	**Hintergrundinformationen**	**129**
10.1	Wurzeln	129
10.1.1	Bilderzeugung	129
10.1.2	Simulation	129
10.1.3	Spiele	130
10.1.4	Zeichentrickfilm	130
10.2	Interaktive und programmierbare Animations-Systeme	131
10.2.1	Keyframe-Systeme	131
10.2.2	Programmierbare Systeme	131
10.2.3	Script-Systeme	132
10.3	Simulation und Robotik	133
10.4	Weitere Konzepte	134
10.4.1	Zielorientierte Animation	134
10.4.2	Partikel-Systeme	135
10.5	Körperanimation	136
10.6	Hardware-Aspekte	137
Anhang	**Literaturverzeichnis**	**141**
	Stichwortverzeichnis	**143**

Vorwort

Computer-Animation ist einer der faszinierendsten Bereiche der Computergrafik. Durch schnelles Abspielen von Einzelbildern wird, wie im Film, die Illusion einer fließenden Bewegung erzeugt. So wie ein Bild durch den beim Betrachter hinterlassenen Eindruck mehr ist als eine Anhäufung von Farbpunkten, so ist auch eine Animation mehr als die Menge ihrer Einzelbilder. Das Gehirn des Betrachters assoziiert mit einer Animation mehr Information über eine Szene, als aus den Einzelbildern. So können beispielsweise Tiefeninformationen daraus gewonnen werden, wie sich Objekte bewegen, wie Glanzlichter über Oberflächen laufen, oder wie Schatten wandern. Diese zusätzlichen Informationen machen Animationen plastischer und realistischer als Einzelbilder. Damit wirkt die Animation auch mehr auf den Betrachter.

Der Begriff »Computer-Animation« wird häufig etwas schlampig verwendet, man wirft allzu oft alle Arten von Animationen in einen Topf. Grob kann man zumindest in zwei Kategorien einteilen:

2-D-Animation
Eine 2-D-Animation wird meist mit einem Malprogramm erzeugt, bei dem die Einzelbilder alle einzeln gemalt werden. Manchmal ist es auch möglich, nur einzelne Schlüsselbilder zu malen, und die Zwischenbilder automatisch erzeugen zu lassen. Das Entscheidende ist, daß alle Operationen nur auf Bildebene möglich und damit zweidimensional sind, daher der Name.

3-D-Animation
Im Gegensatz zur 2-D-Animation wird hier dreidimensional gearbeitet. Es werden Objekte im Raum manipuliert. Diese Objekte werden dann zwar durch ein Bilderzeugungsprogramm (beispielsweise einen Raytracer) in eine Bildebene abgebildet, aber das geschieht automatisch. Der Designer arbeitet bei der 3-D-Animation größtenteils dreidimensional. Der »Reflections-Animator« ist beispielsweise ein 3-D-Animationssystem, das es ermöglicht, Szenen aus »Reflections« zu animieren.

Mein besonderer Dank gilt allen Käufern von »Reflections«. Ohne den guten Verkauf des Programms hätte es keine Fortsetzung gegeben und das Experiment, gute Software preisgünstig zu verkaufen, wäre gescheitert. Die vielen Zuschriften und die durchweg konstruktive Kritik waren eine Resonanz, die mich sehr motiviert hat.

Viel Spaß bei Ihrem privaten Computerfilm

1 Überblick

Der »Reflections-Animator« ist ein Erweiterungspaket für das Programmsystem »Reflections«. Ohne die Basisversion ist es unmöglich, dem »Animator« sinnvolle Ergebnisse zu entlocken. Während bei »Reflections« das Erzeugen von dreidimensionalen Szenen und die anschließende, fotorealistische Bildberechnung im Vordergrund steht, produziert der »Animator« aus diesen Vorlagen einen Trickfilm. Da ein Film aus sehr vielen einzelnen Bildern bestehen kann, wächst die Berechnungszeit mit »Beams« eventuell ins Unermeßliche. Aus diesem Grund hat der »Animator« eine Alternative auf Lager: »Scan« berechnet Bilder schneller, wenn auch mit Abstrichen an der Qualität – die Bilder haben aufgrund der geringeren Berechnungstiefe keine Spiegeleffekte, auch Schatteneffekte sind in der vorliegenden Version noch nicht realisiert.

Wie »Reflections« besteht auch das Animationssystem aus mehreren Teilprogrammen, die unterschiedliche Aufgaben übernehmen:

Animator
Der »Animator« ist das Kernstück des Programmsystems. Er dient zur Definition des Bewegungsablaufes von Körpern und ist ebenso reich an Funktionen wie »Construct«. Auch in seiner Bedienung ist er ähnlich wie »Construct« aufgebaut.

Scan
Für Probeberechnungen braucht man oft nicht die fotorealistische Qualität von »Beams«, hier lohnt es sich, statt dessen »Scan« zu benutzen. Dieses Programm beruht auf dem sogenannten Scanline-Kohärenz-Algorithmus, der im Handbuch zu »Reflections« kurz beschrieben ist. Es liefert im Prinzip die gleichen Resultate wie »Beams«, bei Raytrace-Tiefe 0 und ohne Schatten. Es kann also keine Spiegelung, Brechung und keinen Schlagschatten berechnen. Dafür ist »Scan« wesentlich schneller als »Beams«, was gerade bei der Menge an Bildern, die gelegentlich für eine Animation anfallen, hilfreich sein kann.

Packer:
Der »Packer« übernimmt die Aufgabe, aufeinander folgende IFF-Bilder einer Animation, die von »Show« geliefert wurden, so zu komprimieren, daß für jedes Bild nur die Unterschiede zum davorliegenden Bild gespeichert werden. Das ermöglicht überhaupt erst das Abspielen umfangreicher Animationen, ohne sie vorher auf einen Videorecorder zu übertragen.

Play

Dieses Programm spielt eine von »Reflections« erzeugte Animation ab. Es darf mit den Animationen weitergegeben werden. Dadurch können Sie Ihre Werke auch anderen zugänglich machen, selbst wenn diese den »Reflections-Animator« nicht besitzen sollten.

Make_Anim

Dieses Programm koordiniert während der Berechnung der Animation die Programme, die an der Filmerzeugung beteiligt sind, nämlich »Animator«, »Scan«, »Beams«, »Grid«, »Show« und »Packer«.

Manager

Er entspricht dem Manager aus »Reflections«, mit dem Unterschied, daß die Menüs jetzt auch die Programmteile des »Animator« (einschließlich »Scan« und »Play«) enthalten.

Das Zusammenstellen der Animation übernimmt Make_Anim. Haben Sie im »Animator« alle Parameter festgelegt, können Sie sich erstmal zur Ruhe setzen (oder den Amiga anderweitig nutzen, falls der Speicher reicht). Die Animation wird erzeugt, indem für jedes einzelne Bild folgendes Schema angewendet wird:

1. Der »Animator« erzeugt für jedes Bild eine aktuelle Szenendatei im Construct-Format, in der die bewegten Objekte zum aktuellen Zeitpunkt gespeichert sind.
2. Mit Hilfe von »Scan« oder (»Beams«/ »Grid«) wird daraus ein RGB-Bild der Szene erzeugt.
3. »Show« wandelt das Bild in ein IFF-Bild um.
4. »Packer« berechnet für jedes Bild jeweils den Unterschied zum letzten Bild der Animation und speichert diese Informationen in der Animationsdatei.

Dabei verbraucht der »Reflections-Animator« die meiste Zeit bei der Bildberechnung. Der übrige Aufwand fällt zeitmäßig kaum ins Gewicht.

Wieviel Zeit man zur Erzeugung einer Animation braucht, hängt vom Umfang der Szene, der Größe und Qualität des Bildes, den gewünschten Effekten etc., ab. Der erfahrene Reflections-Benutzer kennt sicherlich die Rechenzeiten von »Beams«. Für eine Raytrace-Animation braucht man bei einer nicht allzu umfangreichen Szene (»Beams«-Rechenzeit mit 2 Stunden pro Bild geschätzt), für eine Sequenz von 50 Bildern, ungefähr eine Woche bis der Film fertig berechnet ist. Wenn Sie mit »Scan« rechnen, bekommen Sie dieselbe Animation in ein bis zwei Tagen, dafür ist dann die Bildqualität nicht ganz so brillant. (Bei Verwendung einer 68020/68881-Turbokarte geschieht das Ganze in beiden Fällen etwa mit vierfacher Geschwindigkeit.) Ein Kompromiß wäre eine lange Animation, bei der der überwiegende Teil mit »Scan« berechnet wird, und nur bestimmte Teile (die möglicherweise besonders knallige Effekte enthalten) mit »Beams«.

Eine weitere wichtige Sache ist der Speicherverbrauch für eine Animation. Da die

Bilder sehr schnell abgespielt werden, müssen sie alle im Hauptspeicher sein, ein Abspielen von Diskette wäre viel zu langsam, selbst eine handelsübliche Festplatte ist noch immer nicht schnell genug. Die Größe einer Animationsdatei hängt von der Anzahl der Bilder ab, aber auch davon, wieviel Bildinformation sich von Bild zu Bild verändert. Es ist selbstverständlich am besten, große Speichermengen zur Verfügung zu haben. Das absolute Minimum liegt bei 1 Mbyte. Damit kann man Animationen mit maximal 200 Bildern erzeugen und auch abspielen – vorausgesetzt die Änderungen pro Bild sind relativ gering. Bei Kamera-Flügen dicht über einer Textur wird beispielsweise sehr viel Speicher verbraucht, da jedes Bild völlig anders aussieht als das letzte. Dabei kann es mit nur einem Mbyte schon bei 20 bis 30 Bildern eng werden – das sind nur wenige Sekunden Film.

1.1 Einige wichtige Begriffe

Der »Animator« orientiert sich stark an der Bedienung von Reflections. Es ist daher sinnvoll, sich zunächst mit der Bedienung dieses Programms vertraut zu machen – im Handbuch zu Reflections ist diese ausführlich beschrieben.

»Reflections-Animator« arbeitet intensiv mit der Körperstruktur von »Construct«. Im »Animator« ist die Körperstruktur sogar noch wichtiger. Denn während »Construct« die Struktur lediglich als Hilfsmittel zum bequemen Konstruieren und Manipulieren von Objekten benötigt, beeinflußt sie beim »Animator« unmittelbar die Bewegungsmöglichkeiten.

Daneben gibt es noch vier wichtige Begriffe, die ab Kapitel 3.2 ausführlich behandelt werden, deren Kenntnis aber für das Verstehen der folgenden Seiten unerläßlich sind. Hier ist ein kurzer Überblick:

1.1.1 Koordinatensysteme

Jeder Körper erhält im »Animator« ein eigenes Koordinatensystem zugeteilt. In »Construct« existiert nur ein globales Koordinatensystem, das Weltkoordinatensystem, das dort für alle Körper gilt. Das bereitet aber Schwierigkeiten, wenn Körper, die bereits schief orientiert im Raum liegen, entsprechend ihrer Orientierung bewegt oder gedreht werden sollen. Im »Animator« kann in diesem Fall auf das körpereigene Koordinatensystem umgeschaltet werden und die entsprechende Bewegung oder Drehung bezieht sich dann auf dieses.

1.1.2 Baum

Im »Reflections«-Handbuch wird im Zusammenhang mit »Beams« der Begriff des Baums erläutert. Im »Animator« spielt der Körperbaum eine zentrale Rolle. Er enthält die Beziehungen aller Körper einer Szene zueinander. Während bei »Construct« die Abhängigkeiten einzelner Körper grundsätzlich beliebig definiert werden konnten,

müssen hier alle Körper eine hierarchische Beziehung haben. Das Koordinatensystem eines Körpers bezieht sich immer auf das seines übergeordneten Objekts. Daraus ergeben sich feste Regeln bei der Bewegungsdefinition. Kapitel 3.2 beschreibt diese genauer.

1.1.3 Skelett

Im »Animator« erhält jeder Körper ein Skelett. Das ist eine vereinfachte Darstellung des Körpers. Zu Beginn ist das Skelett sogar sehr vereinfacht, denn der »Animator« gibt neuen Körpern lediglich eine Linie, einen Würfel, oder ein Viereck als Skelett. Wenn der Anwender am Bildschirm einen Körper bewegt oder dreht, würde das Neuberechnen und -zeichnen aller Körper zu viel Zeit in Anspruch nehmen. »Construct« behilft sich damit, daß nur ein Rahmen bewegt wird, und der Körper erst neu gezeichnet wird, wenn der Anwender diesen Vorgang abgeschlossen hat. Das ist beim »Animator« aber nicht sinnvoll, weil Körper für eine Animation sehr exakt positioniert werden müssen. Man muß während des Verschiebens wenigstens in etwa sehen, wie der Körper an der Stelle aussieht. Statt des ganzen Körpers oder einer Box wird das Skelett des Körpers gezeichnet. Dieses zu verschieben, ist fast in Echtzeit möglich, da ein Skelett nur aus wenigen Linien besteht (beispielsweise ein Strichmännchen für einen Roboter). Das Skelett sollte natürlich die Gestalt des Körpers gut wiedergeben. Der Benutzer hat hierfür die Möglichkeit, das Skelett jedes Körpers weitgehend selbst zu definieren.

1.1.4 Gelenke

Gelenke vereinfachen komplexe Bewegungen von zusammengesetzten Objekten. Ohne dieses Konzept müßte der Anwender für jede Bewegung beispielsweise eines Roboters dessen Rumpf, Kopf, rechten Arm, linken Arm, das rechte und das linke Bein einzeln an die neue Position bewegen. Dreht man beispielsweise einen Oberschenkel um 90 Grad, muß der Unterschenkel so nachgeführt werden, daß er nachher wieder richtig mit dem Oberschenkel zusammenpaßt. Das ist extrem umständlich und außerdem noch ungenau. Wer einen halbwegs realistisch wirkenden Schritt oder gar mehrere auf diese Weise definiert, hat hinterher eine leise Ahnung davon, welch immense Arbeit das menschliche Kleinhirn Tag für Tag leistet. Im »Animator« kann man, um sich diese Arbeit zu ersparen, zwischen einzelnen Körpern Gelenke setzen. Bei einem Roboter wäre etwa ein Gelenk zwischen Oberschenkel und Unterschenkel an der Position des Knies sinnvoll, und eines zwischen Unterschenkel und dem Fuß, zwischen Oberschenkel und Rumpf etc. Bewegt man jetzt den Fuß, so werden Unterschenkel und Oberschenkel mitgedreht, was eine Menge Arbeit erspart. Ein Schritt ist so in wenigen Sekunden ausgeführt. Der »Animator« kann anhand der Gelenke immer feststellen, welche Körperteile von einer Bewegung betroffen sind, und wieweit sie verändert werden müssen, um diese Bewegung mitzumachen. Übrigens: Das menschliche Kleinhirn vereinfacht sich die Arbeit auf ähnliche Weise.

2 Einführung

2.1 Kopieren von »Reflections-Animator«

Bevor Sie anfangen, mit dem »Reflections-Animator« zu arbeiten, sollten Sie sich eine Kopie der Originaldisketten anfertigen. Danach legen Sie das Original an einen sicheren Ort und arbeiten nur noch mit den Kopien. Denken Sie beim Aufbewahren Ihrer Originale stets daran: Disketten vertragen weder Hitze noch Strahlung, besonders üble Datenkiller sind Magnetfelder – beispielsweise in der unmittelbaren Nähe von Computermonitoren.

2.2 Installieren der neuen Programme

»Reflections-Animator« wird mit der Basisversion von »Reflections« zu einem System vereinigt. Die einzelnen Komponenten sind – wie bei »Reflections« – Einzelprogramme. Diese müssen aber, um vom Manager erkannt zu werden, in den richtigen Ordnern liegen.

Dorthin werden sie von einem CLI-Skript auf der Programmdiskette gebracht. Es gibt zwei verschiedene Dateien für zwei Ausgangssituationen.

2.2.1 Installieren auf Festplatte

In diesem Fall müssen lediglich die neuen Programme zu den bereits existierenden kopiert werden. Geben Sie dazu in der Shell oder im CLI folgenden Zeile ein:

```
refl_anim:a_to_hard
```

Damit werden alle benötigten Dateien in das logische Gerät REF: kopiert, in dem schon alle »Reflections«-Dateien liegen – Voraussetzung ist allerdings, daß »Reflections« bereits korrekt installiert ist; wenn nicht, sollten Sie das zunächst nachholen.

2.2.2 Installieren auf zwei Disketten

Auf einer einzigen Diskette ist leider nicht genügend Platz, um sowohl »Reflections« als auch den »Animator« aufzunehmen. Daher müssen die Programme auf zwei Disketten aufgeteilt werden. Sie haben dann eine »Reflections«-Diskette wie bisher, die zum Konstruieren von 3-D-Szenen verwendet wird, und eine zweite zum Erzeugen von Animationen.

Fertigen Sie zunächst eine Kopie Ihrer »Reflections«-Diskette an. Das Installationsprogramm wird diese Kopie so verändern, daß genügend Platz für den »Animator« übrig ist. Verwenden Sie dafür unbedingt eine Kopie Ihrer Arbeitsdiskette. Geben Sie dann in der Shell oder im CLI folgende Zeile ein:

```
refl_anim:a_to_disk
```

Diese Skript-Datei löscht auf der »Reflections«-Diskette alles, was dort nicht mehr gebraucht wird (beispielsweise »Construct«, »Show« und Get_IFF) und kopiert dann die »Animator«-Programme an deren Stelle.

Wenn Sie bei einem Zwei-Disketten-System mit »Construct« eine Szene konstruiert haben, die Sie animieren wollen, so kopieren Sie dieselbe in den »Szenen«-Ordner der Diskette mit dem »Animator«. Dort erzeugen Sie dann die Animation.

3 Animator

3.1 Überblick

Der »Animator« ist ein Editor für 3-D-Animationen. Das geschieht mittels der sogenannten Key-Frame-Technik. Statt mühsam jedes einzelne Bild zu konstruieren, bewegt man einfach einen Körper an seine neue Position, legt den Zeitpunkt fest, an dem er diese Position erreichen soll – fertig. Durch Interpolation errechnet der »Animator« sämtliche Zwischenpositionen aller Körper in einer Szene. Durch Angabe weniger Schlüsselpositionen sind so schon komplexe Bewegungsabläufe möglich.

Bewegungen von Körpern werden zu Sequenzen zusammengefaßt. Eine Sequenz enthält für einen Körper eine Anzahl von Schlüsselpositionen samt zugehörigen Zeitpunkten. Sequenzen können durch Einfügen neuer Positionen oder Löschen von Positionen editiert werden. Oder deren Zeitpunkte können auf der Zeitachse verschoben werden. Alle diese Aktionen bewirken eine Änderung des Pfades, der Bewegungsbahn des Körpers. Der Anwender kann schon während der Konstruktionsphase eine Sequenz, oder nur Teile daraus, als Skelett-Animation betrachten, um einen schnellen Überblick über den Bewegungsablauf zu bekommen und gegebenfalls noch vor der aufwendigen Bildberechnung Korrekturen einzubauen.

Die Manipulationen der Objekte werden am Bildschirm an den Skeletten der Körper ausgeführt. Ein Skelett ist die vereinfachte Darstellung des Körpers (beispielsweise als Strichmännchen). Es besteht nur aus wenigen Linien, die dem Betrachter die ungefähre Form des Körpers verdeutlichen. Damit ist annähernd eine Echtzeitbewegung von Körpern am Bildschirm möglich. Selbstverständlich kann jeder Körper per Knopfdruck so dargestellt werden, wie von »Construct« her gewohnt.

Eine Skelett-Animation ist damit in wenigen Sekunden berechnet, so daß ein schnelles Feedback ermöglicht wird. Die nächste Stufe ist die Wire-Frame-Animation. Dabei wird der Körper exakt in den späteren Konturen dargestellt. Lediglich die Oberfläche wird nicht in die Berechnung einbezogen, die Körper erscheinen daher durchsichtig. Hierfür sind einige Minuten als Rechenzeit nötig.

Das Nonplusultra ist natürlich die Raytrace-Animation. Daß diese erheblich länger dauert, muß nicht betont werden. Der »Animator« liefert dazu lediglich die Bewegungsinformation für die Körper einer Szene und ermöglicht die Interpolation – die Berechnung der Zwischenpositionen der Körper. Wie daraus dann die Bilder

berechnet werden, interessiert hier nur am Rande, denn diese Aufgabe übernimmt ein weiteres Teilprogramm, das im Kapitel 4 beschrieben wird: »Make_Anim«.

Nach erledigter Bewegungsdefinition wird diese in einer sogenannten Sequenz-Datei abgelegt. Aus dieser Datei, der Originalszene plus weiteren Informationen, wie etwa Anzahl Bilder, Bildgröße und -qualität, Art der Bildberechnung etc., bezieht »Make_Anim« den Auftrag für eine Animation. Dieses Programm wiederum läßt sich, vom »Animator« wird Bild für Bild das Aussehen der Körper interpolieren, von »Beams« oder »Scan« die Bilder berechnen und vom Packer zu einer Animation zusammenpacken. Dies ist dann allerdings ein Vorgang, der mehrere Tage dauern kann.

3.2 Die Fenster des »Animator«

Nach dem Start des »Animator« sind – ähnlich wie bei »Construct« drei Fenster am Bildschirm zu erkennen, die für die Bedienung des »Animator« eine unterschiedliche Bedeutung haben.

3.2.1 Das Plotfenster

Das Plotfenster ist das größte Fenster. In diesem werden alle grafischen Ausgaben angezeigt, hier erledigt der Benutzer alle Manipulationen an Körpern mit Hilfe der Maus. Neu sind die Gadgets, die den eigentlichen Plotbereich umgeben. Ein Teil davon dient zur geometrischen Manipulation von Körpern. Diese Funktionen wurden im Gegensatz zu »Construct« vom Menü auf die Gadgets verlegt. Ein anderer Teil zur bequemeren Skalierung des Plot-Bereiches vorgesehen. Der Rest zur Definition der Kamera, ähnlich wie bei »Construct«.

3.2.2 Das Statusfenster

Es entspricht in seinen Funktionen dem Listfenster von »Construct«, allerdings ist es wesentlich kleiner. Das Menü des »Animator« ist mit dem Statusfenster verknüpft. Sollte das Menü eventuell einmal nicht herunterklappen, obwohl die rechte Maustaste gedrückt ist, so hilft meist ein Klick ins Statusfenster, um dieses zu aktivieren. Hier werden auch kurze Meldungen des »Animator« ausgegeben.

3.2.3 Das Sequenzfenster

Mit Hilfe des Sequenzfensters legt der Anwender Schlüsselpositionen und deren Zeitpunkt fest. Das große Feld ist für das Verändern Schlüsselpositionen (und deren Zeitpunkte) vorgesehen. Rechts daneben sind in einem Feld von Gadgets die Funktionen zur Manipulation von Sequenzen zusammengefaßt.

Zusätzlich erscheinen bei Bedarf zeitweilig weitere Fenster. Das Auswahlfenster sieht ebenso aus wie in »Construct«. Es wird analog benutzt – alle Funktionen können im

»Reflections«-Handbuch nachgelesen werden. Außerdem erscheint beim Laden und Speichern ein Dateifenster zur Auswahl von Dateien, oder beim Festlegen der Beleuchtung ein Lichtfenster zur Bestimmung von Lichtfarbe und Helligkeit – diese beiden Fenster werden weiter unten näher erläutert.

3.3 Kurzübersicht Menüs

Alle häufig benutzten Funktionen sind – wie bei »Construct« – neben der Menüleiste auch über die Tastatur erreichbar. Neu ist, daß die entsprechenden Tasten nicht mehr gleichzeitig mit der rechten Amiga-Taste (rechts neben der Space-Taste) gedrückt werden müssen, sondern die Taste alleine genügt. Außerdem sind sämtliche Funktionen ständig über die Tastatur erreichbar, auch wenn das Status-Fenster gerade nicht aktiv ist (einzige Ausnahme: wenn eine Funktion längere Zeit rechnet und dafür die Eingabe blockiert). Eine weitere interessante Neuigkeit gegenüber »Construct« ist, daß immer jeweils das Fenster aktiv ist, über dem sich der Mauszeiger befindet. Das erspart bei Funktionen, die die rechte Maustaste benutzen, das vorherige Aktivieren des Fensters.

Die Menüleiste enthält sieben Punkte mit einer unterschiedlichen Zahl Unterpunkte. Diese sind weiter unten ausführlich beschrieben. Hier folgt eine Kurzübersicht – am besten kopieren Sie diese und hängen sie während der Einarbeitungszeit über den Monitor.

3.3.1 Menüpunkt »Projekt«

Laden und Speichern von Dateien, Programmende.

LoadSkl
Laden von Skelett-Dateien

SaveSkl
Speichern von Skelett-Dateien

LoadSeq
Laden von Sequenz-Dateien

SaveSeq
Speichern der Sequenz

SaveSz
Speichern der augenblicklichen Gesamtszene im »Construct«-Format

Quit
Programmende

3|3|2| Menüpunkt »Plotten«

Bestimmung von Darstellungsparametern (analog zu »Construct«).

Plotkörper
Festlegen der Plotkörper-Liste

Hidd0
Wie bei »Construct«

Hidd1"
Wie bei »Construct«

Hidd2»
Wie bei »Construct«

Gitter
Raster über den Bildschirm legen

vorne
Wie bei »Construct«

hinten
Wie bei »Construct«

rechts
Wie bei »Construct«

links
Wie bei »Construct«

unten
Wie bei »Construct«

oben
Wie bei »Construct«

persp
Wie bei »Construct«

Bild*2
Wie bei »Construct«

Wire-Frame
Darstellung der Plotkörper wie bei »Construct«

3.3.3 Menüpunkt »Körper«

Manipulationen an Körpern (beispielsweise Drehen oder Ändern der Position).

ausrichten
Ausrichten aller Körper auf das Koordinatensystem eines Körpers

Eigenschaften
Manipulation von Körpereigenschaften

clear_eigenschaften
Zurücksetzen aller Körpereigenschaften auf die voreingestellten Werte

Körpernamen
Anzeigen aller Körpernamen der Plotkörper, sowie ihrer Unterkörper

Gelenknamen
Anzeigen der Gelenknamen der aktuellen Sequenz

zentrieren
Bei Paralleldarstellung Bildschirm so ausrichten, daß er auf das Zentrum der Plotkörper gerichtet ist

Lichtfarbe
Einstellen der Farbe einer Lichtquelle

3.3.4 Menüpunkt »Special«

Drei besondere Einstellungen:

int_quick
schnelleres Interpolieren einschalten

int_good
genaueres Interpolieren einschalten

show_gelenk
Darstellen aller Körper, die an einem Gelenk hängen

3.3.5 Menüpunkt »Sequenz«

Funktionen zur Verwaltung von Filmsequenzen.

akt_seq
Einstellen der aktuellen Sequenz

neu_seq
Erzeugen einer neuen Sequenz

Untermenüpunkte zu »neu_seq«:

neu_seq.körper
Sequenz für ein einzelnes Objekt

neu_seq.kamera
Kamerasequenz

neu_seq.licht
Lichtsequenz

Take_Snaps
Übernahme der »Snaps« von einer anderen Sequenz

Gelenke
Festlegen von Gelenken

Untermenüpunkte für »Gelenke«:

Gelenke.neu
Neues Gelenk definieren

Gelenke.bewegen
Gelenk neu positionieren

Gelenke.delete
Gelenk aus der Liste löschen

Gelenke.take
Gelenke einer anderen Sequenz übernehmen

Gelenke.fertig
Gelenkdefinition abschließen

Flags
Einstellen von Sequenzeigenschaften

Untermenüpunkte für »Flags«:

Die »Flags« legen fest, wonach interpoliert werden soll, es gibt vier Möglichkeiten:

Flags.Linear

Flags.Weich

Flags.Ziel

Flags.Richtung

Take
Definition eines »takes«

Untermenüpunkte für »Take«:

Take.zeige
Zeigt alle Sequenzen des aktuellen Takes an

Take.define
Zusammensetzen der Take-Sequenzen

multi_seq
Ein- bzw. Ausschalten des »Multi-Sequenz«-Modus

plot_pfad
Darstellen der Schwerpunktbahn eines Körpers

seq_move
Einschalten des »Sequenz-Move«-Modus

Delete-seq
Löschen von Sequenzen

3.3.6 Menüpunkt »PreView«

Erzeugung von Skelett- oder Wire-Frame-Animationen.

Make_anim.skelett
Erzeugen einer Skelettanimation

Make_anim.Wire-Frame
Erzeugen einer Wire-Frame-Animation

3.3.7 Menüpunkt »Skelett«

Erzeugung und Manipulation von Körperskeletten.

Make_Skelett
Beginn der Skelett-Definition

Box
Kasten als Skelett

Fläche
Flächen als Skelett

Linie
Linie als Skelett

Neu_Pkt
Zusätzlicher Skelettpunkt

Neu_Linie
Zusätzliche Skelettlinie

Bewege_Pkt
Bewegen eines Skelettpunkts

Lösche_Pkt
Löschen eines Skelettpunkts

Lösche_Linie
Löschen einer Skelettlinie

fertig Skelett
Abschließen der Skelett-Definition

3|4| Koordinatensystem und Bäume

Um versiert mit dem »Animator« umzugehen, ist es unerläßlich, einiges über die zugrundeliegenden Datenstrukturen zu wissen. Auch wenn dies eine relativ trockene Sache ist, sollten Sie diese Seiten keinesfalls überblättern – sie ersparen Ihnen spätere Schwierigkeiten.

»Construct« läßt dem Anwender weitgehende Freiheiten in der Definition der Körper-Struktur. Dieser an sich angenehme Umstand hat sich für den »Animator« zunächst als störend herausgestellt, da dieser eine streng hierarchische Struktur braucht.

3|4|1| Koordinatensysteme

Jedes Koordinatensystem besteht aus einem Ursprung (auch Nullpunkt genannt) und einer Anzahl von Vektoren, welche die Grundrichtungen des Koordinatensystems angeben. Diese Vektoren, auch Basisvektoren genannt, können in beliebigen Richtungen liegen. Normalerweise ordnet man sie jedoch senkrecht zueinander an. 2-D-Koordinatensysteme sind mit zwei Basisvektoren eindeutig definiert, das 3-D-System mit drei. Das sogenannte Weltkoordinatensystem des »Reflections-Animator« hat als Ursprungspunkt den absoluten Nullpunkt (dieser muß als einziger nicht angegeben werden) und drei Basisvektoren. Auch die Vektoren werden normalerweise nicht extra angegeben, sondern es wird stillschweigend vorausgesetzt, daß ein Vektor nach hinten, einer nach rechts und einer nach oben zeigt.

Teilweise herrschen unterschiedliche Meinungen darüber, wie die einzelnen Vektoren heißen. Das am häufigsten verwendete Modell nennt die nach rechts zeigende Achse »x«, die nach oben »y« und die nach hinten in den Raum zeigende »z«. Eindeutig ist dagegen die Notation – sie erfolgt in alphabetischer Reihenfolge: Jeweils der erste genannte Punkt ist der x-Wert, der zweite der y-Wert und der dritte der z-Wert. Wenn ein Punkt die Koordinaten (1,5, 10,2, 3,1) hat, bedeutet das, daß er 1,5 Einheiten vom

Nullpunkt in Richtung des x-Vektors entfernt liegt, 10,2 Einheiten in Richtung des y-Vektors und 3,1 Einheiten in Richtung des z-Vektors.

Dreht man dagegen die Basisvektoren um einen beliebigen Winkel, beispielsweise 45 Grad, und bezieht jetzt die Koordinaten eines Objekts auf die veränderten Basisvektoren, dann hat das die gleiche Wirkung, als ob das Objekt selbst gedreht worden wäre. Die Objektkoordinaten sind jedoch nicht verändert worden, sondern nur ihr Bezugssystem. Dadurch kann die interne Struktur des Objekts vollkommen erhalten bleiben. Genau dieses Prinzip wendet der »Animator« an: Beim Drehen, Bewegen oder Vergrößern eines Körpers werden nicht seine Bestandteile verändert, sondern nur sein Koordinatensystem.

Bei von »Construct« übernommenen Körpern könnte jetzt ein Problem auftreten. Denn bei »Construct« ist es möglich, daß dasselbe Objekt zu mehreren Körpern gehört. Wenn jetzt – wie beim »Animator« – jeder Körper sein eigenes Koordinatensystem hat, könnte eine Drehung des einen Körpers ein Objekt bewegen, das aber gleichzeitig zu einem anderen Körper gehört, der sich möglicherweise in eine andere Richtung bewegt. Beispielsweise könnte ein Körper »Glas«, der aus einer Menge von Dreiecken aufgebaut ist, Objekte enthalten, die gleichzeitig zum Körper »Tisch-Gegenstände« gehören, und außerdem zum Körper »Glas-Objekte«. Das ist für »Construct« keine außergewöhnliche Konfiguration (Bild 1).

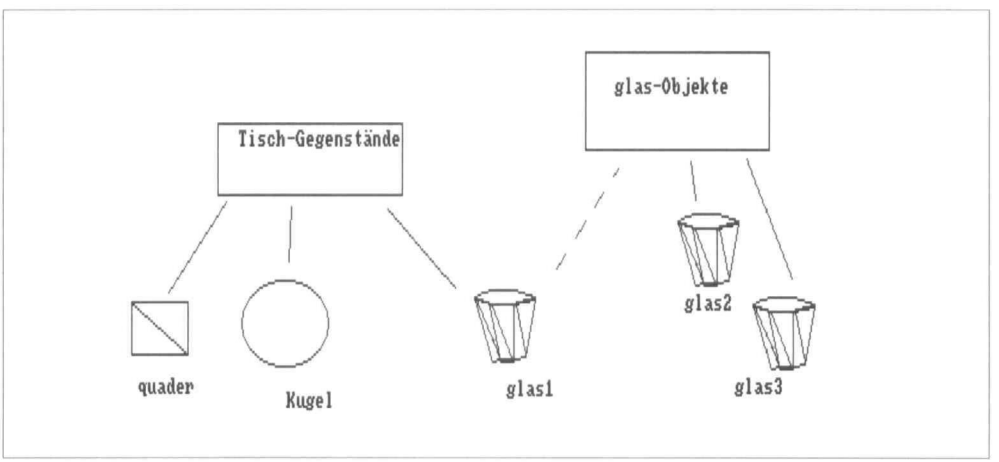


Bild 1: Ein Körper darf nur einen Vater haben

Für »Construct« bedeutet das kein Problem, da es nur »Stilleben« behandelt – Bewegung kommt nicht vor. Der »Animator« dagegen verpaßt jedem Körper ein Koordinatensystem, das für die Ausrichtung der Teilelemente zuständig ist. Genau an diesem Punkt fangen die Probleme an. Aus diesem Grund ist es unerläßlich, die Freiheit von »Construct« etwas einzuschränken. Die einfachste Möglichkeit wäre, ganz

strikt vorzuschreiben, daß jedes Objekt nur zu einem Körper gehören darf. Das wäre aber eine unnötige Quelle für Inkompatibilitäten. Es genügt schon, wenn man eine Baumstruktur einführt. Diese Technik ist auch unter dem Begriff Vater-Sohn-Beziehung bekannt. Dabei gibt es zwischen Körpern fest definierte Beziehungen:

Vater
Ein Körper »A« ist Vater eines Körpers »B«, wenn alle Objekte von »B« auch Objekte von »A« sind.

Sohn
»A« ist Sohn von »B« wenn alle Objekte von »A« auch Objekte von »B« sind.

Dazu kann man noch weitere Beziehungen einführen:

Bruder
»A« ist Bruder von »B«, wenn »A« und »B« den gleichen Vater haben.

Verwandte
»A« gilt als verwandt zu »B«, wenn ein Teil der Objekte von »B« auch Objekte von »A« sind.

Dazu müssen bei der Konstruktion der Körper einige Regeln eingehalten werden:

- Jeder Körper muß genau einen Vater haben (einzige Ausnahme: der Körper »all«, der als Vater aller anderen Körper gilt).

- Zwei Körper dürfen nur verwandt sein, wenn sie auch Brüder sind (also den gleichen Vater haben) und selbst keine Söhne mehr haben.

- Gehört ein Dreieck (oder eine Kugel) zu mehreren Körpern, so gilt für seine Koordinaten nur der Körper, der keine Söhne mehr hat. Gehört es zu mehreren Körpern, die keine Söhne mehr haben, ist das nur erlaubt, wenn diese Körper Brüder sind.

Bild 2 zeigt einen Baum, der durch die Körper »A«, »B«, »C«, »D«, »E«, »F«, »G«, »H«, »AB«, »CDE«, »FGH«, »CDE-FGH« und »AB-CDE-FGH« erzeugt wird. Bereits die Namensgebung verdeutlicht, welche Objekte die einzelnen Körper enthalten. Dennoch lohnt sich ein Blick auf die einzelnen Körper, um die Abhängigkeiten zu verdeutlichen:

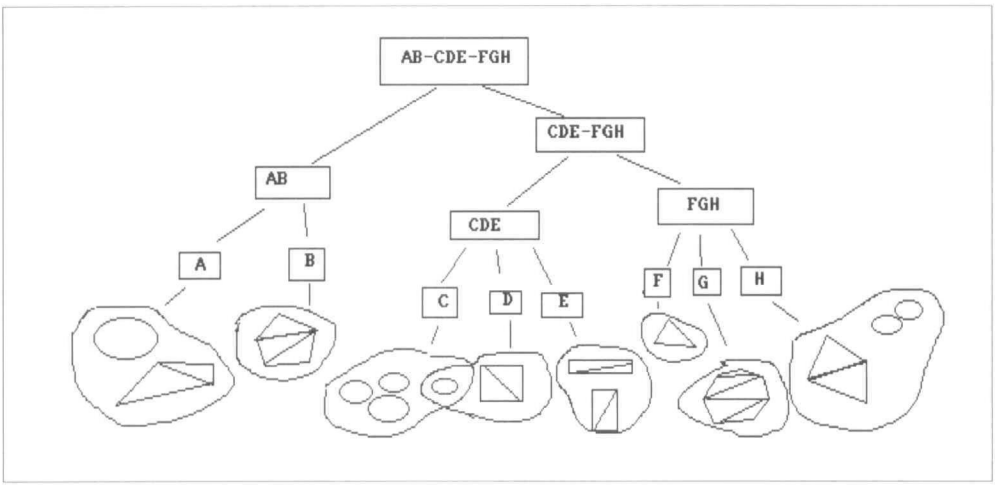


Bild 2: Die Objekthierarchie des »Reflections-Animator«

■ Die Körper »A bis H« haben selbst keine Söhne mehr, folglich gehören alle enthaltenen Objekte ausschließlich zu diesen.

■ Körper »A« enthält beispielsweise eine Kugel und zwei Dreiecke. Diese Objekte gehören (für »Construct«) auch zu »AB« und »AB-CDE-FGH«. Im »Animator« sind sie dagegen nur zu »A« zugeordnet. Damit gilt nur das Koordinatensystem von »A« für diese Objekte.

■ »A« und »B« sind Brüder, da »AB« ihr gemeinsamer Vater ist.

■ »B« und »C« sind keine Brüder weil sie verschiedene Väter (»AB« und »CDE«) haben.

■ »CDE« ist Vater von »C«, »D« und »E«. »CDE-FGH« ist nicht Vater von »C« (allenfalls Großvater, wenn diese Beziehung existieren würde) sondern von »CDE«.

■ »C« und »D« sind Brüder und gleichzeitig Verwandte, denn sie enthalten die gleiche Kugel als Objekt. Das ist nur erlaubt, weil sie den gleichen Vater »CDE« haben.

3.4.2 Erzeugung des Körperbaums

Nachdem in »Construct« keine hierarchische Objektstruktur vorgesehen war, muß der »Animator« diese Struktur zurückberechnen. Angenommen, man würde mit »Construct« die Körper »A« und »B« mittels des Menüs »Addneu« zu einem neuen Körper verbinden, so entsteht dabei nichts anderes als »AB«. Alle Körper die mittels »Addneu« zu einem neuen vereinigt werden, gelten als Brüder sind gleichzeitig Söhne des neu erzeugten Körpers. Jeder Körper, der mit »Addneu« erzeugt wurde, gilt als Vater der Körper, aus denen er zusammengesetzt ist.

Alle anderen Körper einer »Construct«-Datei, die aus Design-Gründen erzeugt wurden, und die nicht in den Körperbaum passen, müssen gelöscht werden (nur die Körper, nicht deren Objekte).

Selbstverständlich ist es schwierig, sich den Überblick zu verschaffen, welche Körper in einer Szene gerade Väter, Söhne oder Brüder sind. Beim Einlesen einer Szene baut der »Animator« automatisch den Körperbaum auf. Er bezieht die Informationen dafür aus den Objektlisten der Körper. Dann vergleicht er alle Körper auf ihre »Verwandtschaftsbeziehungen«. Findet er Körper, welche die oben genannten Regeln verletzen, nimmt er die Szene nicht an. In diesem Fall bleibt nichts anderes übrig, als die Szene mit »Construct« neu zu strukturieren. Um diesen Vorgang zu erleichtern, gibt »Animator« eine Diagnose-Datei (unter dem Namen »ram:diagnose«) aus. In dieser steht für jeden Körper der Vater, die Brüder, Söhne und gleiche Körper, sowie ein Kommentar, warum er vom »Animator« nicht übernommen werden konnte.

3|5| Einlesen von »Construct«-Szenen

Ein Film, in dem nichts zu sehen ist, könnte ziemlich fad werden. Nachdem der »Animator« keine eigenen Objekte konstruieren kann, muß er diese aus einem anderen Programm importieren. Dies geht nur mit »Construct«, da der »Animator« die Strukturinformationen benötigt. Wer lieber mit anderen Modellierprogrammen arbeitet, kann die Szenen mit einem der Tools von »Reflections« so konvertieren, daß sie von »Construct« gelesen werden können. Dieser Umweg ist dringend erforderlich, da dem »Animator« sonst die Strukturinformationen fehlen. Der »Animator« liest beim Programmstart automatisch die vom Manager angegebene Szenendatei ein.

HINWEIS: Es besteht keine Möglichkeit, weitere Szenendateien nachzuladen, wie von »Construct« her gewohnt.

Aus diesem Grund sollte man sich einige Regeln einprägen, die bei der Arbeit mit »Construct« und dem »Animator« hilfreich sind:

- Stellen Sie in »Construct« alle Objekte und Körper zusammen, die in der Animation vorkommen sollen. Speichern Sie diese in einer einzigen Datei. Dies ist die sogenannte Grunddatei für den »Animator«, die er nach dem Start liest.

- Löschen Sie alle überflüssigen Körperbeschreibungen in »Construct«. Sie brauchen nur die Körperbeschreibungen für die Körper, die Sie im »Animator« manipulieren wollen. Darüber hinausgehende Beschreibungen erhöhen nur den Rechenaufwand für den »Animator«. Objekte, die zwar in der Szene vorkommen, aber nicht bewegt werden sollen, gehören am sinnvollsten nur zu »all«.

- Der »Animator« nimmt keine Szenen mit unkorrekter Baumstruktur an. Die Fehlergründe finden Sie dann in der Datei »ram:diagnose«.

■ Überlegen Sie sich schon bei der Konstruktion der Grunddatei genau, welche Körper im »Animator« wie bewegt werden sollen.

■ Soll ein Körper stets nur als Ganzes bewegt werden, genügt eine einzige Körperbeschreibung.

■ Sollen beim Körper die einzelnen Bestandteile unabhängig oder gekoppelt bewegt werden, braucht man für jedes Bestandteil eine eigene Körperbeschreibung. *BEISPIEL:* Sie haben in »Construct« einen Vogel modelliert und wollen ihn im »Animator« fliegen lassen. Wenn er nur als Ganzes dahingleiten soll, genügt ein Körper »Vogel«. Soll er aber noch mit den Flügeln schlagen, den Kopf drehen etc., so muß jeder Bestandteil in einem eigenen Körper stecken. Sie müssen für diesen Fall also die Körper »R-Flügel«, »L-Flügel«, »Rumpf«, »Kopf« etc. getrennt benennen.

■ Speichern Sie in »Construct« alle Körper in einer definierten Grundhaltung und in einer Grundrichtung orientiert. Diese Grundrichtung übernimmt der »Animator« als die Ost-West-Richtung. Orientieren Sie also generell alle Körper, bei denen eine Vorzugsrichtung existiert, in Ost- oder West-Richtung. Sie müssen diese Regel zwar nicht einhalten, aber es erleichtert die Handhabung.

■ Wenn Sie einen Körper modelliert haben, der später weiter verwendet werden soll, speichern Sie ihn in seiner Grundorientierung ab (eventuell in einer eigenen Datei). Wenn Sie den Körper in mehreren verschiedenen Animationen verwenden wollen, verändern Sie ihn nicht mehr, bewegen Sie ihn nicht, drehen, vergrößern oder verzerren Sie ihn nicht. Laden Sie ihn vielmehr in »Construct« zu Ihrer Szene dazu und speichern ihn unverändert in der Grunddatei, auch wenn er völlig falsch liegt oder zu klein ist. Das Bewegen geht mit dem »Animator« viel einfacher. Ist der Körper unverändert übernommen, so können Sie Bewegungssequenzen, die für diesen Körper zuvor erstellt wurden, hier erneut übernehmen.

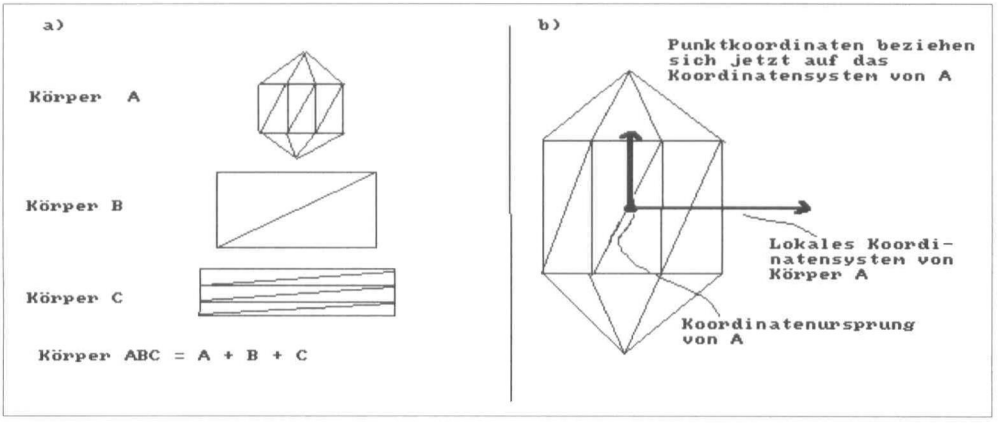


Bild 3: Beziehungen zwischen Körpern

Beim Einlesen einer Szene führt der »Animator« zunächst einige Vorarbeiten aus. Als Beispiel lohnt es sich, die Szene aus Bild 3, Teil a) näher anzusehen. Sie enthält die Körper »A«, »B«, »C« und »ABC«, die die Vereinigung von »A«, »B« und »C« darstellen.

Der »Animator« vergleicht zunächst alle Körper miteinander, um Überschneidungen festzustellen. Bei »A«, »B« und »C« findet er nichts, stellt aber fest, daß alle drei Teile von »ABC« sind. Somit erzeugt er einen Baum, mit »ABC« als Vater und »A«, »B« und »C« als Söhne.

Als nächstes erzeugt das Programm für jeden Körper ein eigenes Koordinatensystem. In Bild 4 ist dies für Körper »A« gezeigt. Er setzt den Ursprung (oder Nullpunkt) des Koordinatensystems in den Mittelpunkt des Körpers. Die Koordinatenachsen zeigen nach rechts, oben und hinten (deswegen ist es auch am günstigsten, wenn alle Körper achsenparallel ausgerichtet sind). Außerdem rechnet »Animator« jetzt die Koordinaten der Punkte jedes Körperobjekts auf das Koordinatensystem von »A« um.

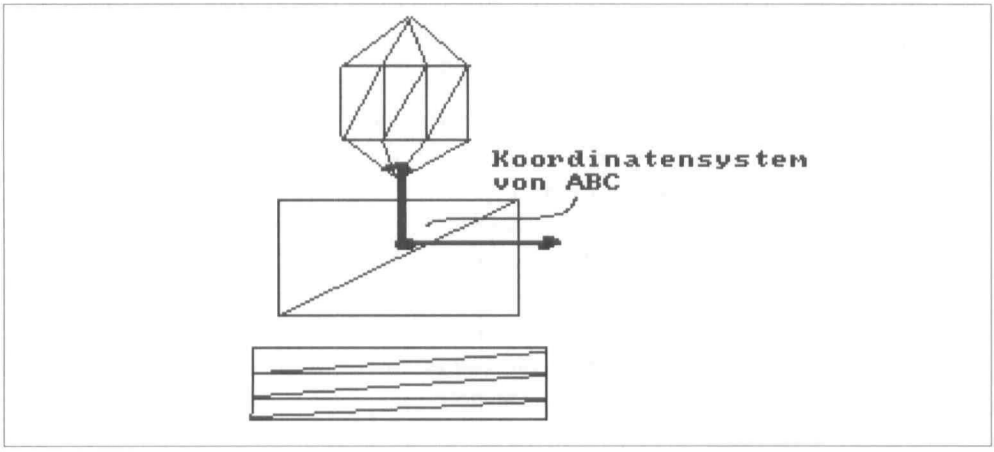


Bild 4: Das Koordinatensystem von A

Danach berechnet das Programm den Körperbaum. »ABC« erhält ebenso ein Koordinatensystem, das in den Mittelpunkt dieses zusammengesetzten Körpers gelegt wird (siehe Bild 4). Die Körperobjekte werden jetzt aber nicht mehr neu bearbeitet – sie sind bereits durch die Koordinaten der Söhne eindeutig festgelegt. Dagegen werden jedoch die Koordinatensysteme von »A«, »B« und »C« jetzt so umgerechnet, daß sie sich auf »ABC« beziehen. Für sie ist von jetzt an der Mittelpunkt des übergeordneten Körpers »ABC« der Nullpunkt.

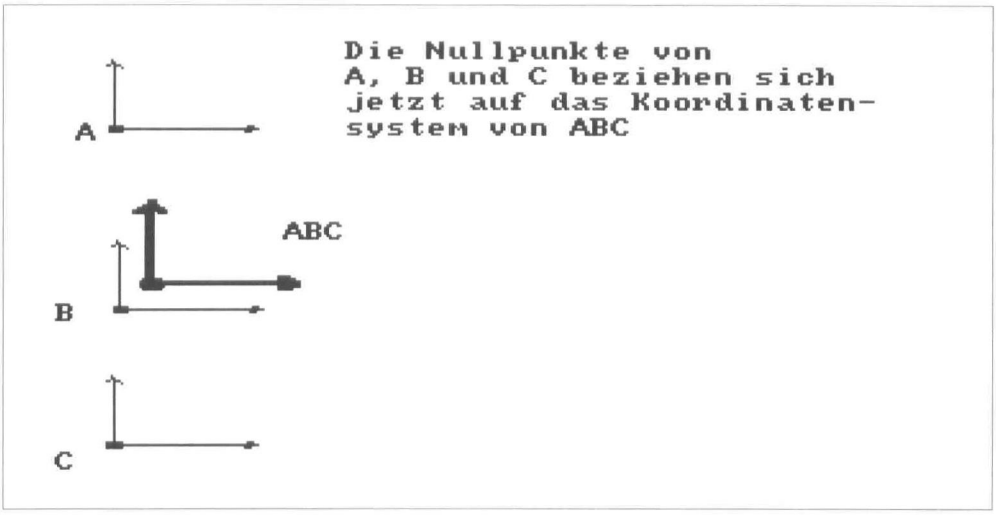


Bild 5: Koordinaten eines zusammengesetzten Körpers

Wenn im »Animator« ein Körper gedreht oder bewegt wird, beeinflußt das nur sein Koordinatensystem. Bei einer Bewegung wird sein Koordinatenursprung bewegt, bei einer Drehung werden nur die Achsen gedreht. Die eigentlichen Objekte müssen hierfür nicht neu berechnet werden, da sich ihre Koordinaten auf das bereits bearbeitete Koordinatensystem beziehen (Bild 5 und 6).

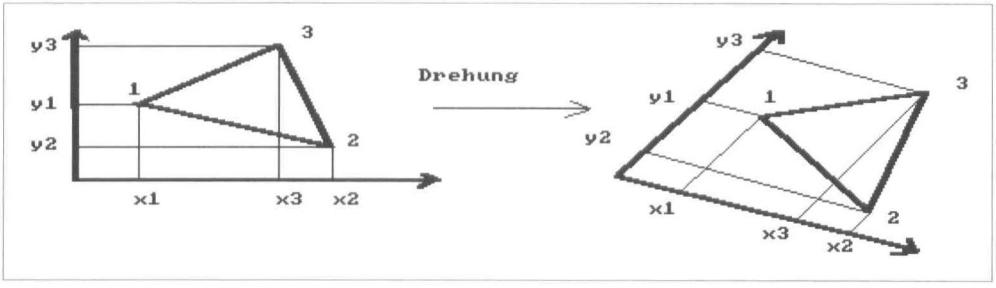


Bild 6: Bei der Drehung eines Körpers dreht sich das gesamte Koordinatensystem

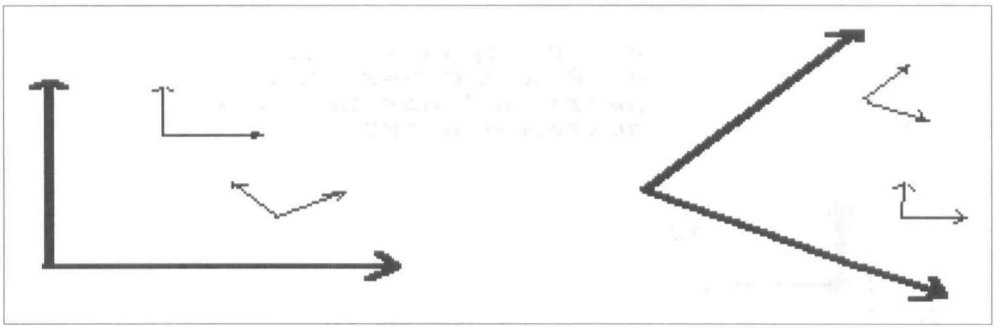


Bild 7: Eventuell enthaltene Körper werden mitgedreht

Erst wenn es darum geht, die Objekte in einer fertiggestellten Szene darzustellen, werden sie ins Weltkoordinatensystem umgerechnet.

Verdoppelte Körper

Aus »Construct« ist das Verdoppeln von Körpern bekannt. Damit ist es schnell möglich, umfangreiche Szenen zu erzeugen. »Construct« erzeugt allerdings beim Verdoppeln nur neue Grundobjekte (Dreiecke und Kugeln). Eventuell ebenfalls betroffene Körperbeschreibungen werden nicht verdoppelt.

Beispielsweise könnte ein verdoppelter Roboter mit vielen Teilkörpern (Arme, Beine etc.) Probleme bereiten, da für seinen Doppelgänger die Teilkörper nicht existieren. Diese müßten dann neu geschaffen werden. Es gibt aber mit »Construct« noch einen weiteren Weg, Körper zu verdoppeln, der für diesen Fall eher zu empfehlen ist, wenn er auch etwas umständlicher ist.

Nachdem der Körper erzeugt ist, wird er in einer eigenen Datei gespeichert (eventuelle überflüssige Körper sollten vorher gelöscht werden). Zum Verdoppeln wird er einfach neu geladen. Dies hat gegenüber der Funktion »doppeln« den Vorteil, daß alle Abhängigkeiten erhalten bleiben – es wird also ein echter Doppelgänger erzeugt. Der Nachteil für die Arbeit mit »Construct«: Die auf diesem Weg verdoppelten Körper haben alle den gleichen Namen, wie ihre Ursprungskörper, was zu einiger Verwirrung führen könnte.

Auf diesen Fall ist der »Animator« dagegen schon eingerichtet. Sobald er auf Körpernamen stößt, die bereits existieren, fügt er einfach eine »/1« an. Oder falls einer sogar mehrfach existiert »/2« oder »/3« usw.

BEISPIEL: Sie haben mit »Construct« ein Flugzeug geschaffen. Es besteht aus den Teilkörpern »Rumpf«, »Rechter_Flügel«, »Linker_Flügel«, »Leitwerk« und »Höhenruder«. Dieses Flugzeug haben Sie dann in einer eigenen Datei gespeichert. Jetzt

möchten Sie eine Animation mit zwei dieser Flugzeuge erzeugen. Um das zu erreichen, laden Sie in »Construct« die Flugzeug-Datei zweimal (damit haben Sie das Flugzeug und jeden seiner Körper doppelt), und speichern das Ganze wieder erneut. Wenn ANIMATOR diese Szene liest, erzeugt er folgende Körpernamen: »Rumpf«, »Rechter_Flügel«, »Linker_Flügel«, »Leitwerk«, »Höhenruder«, »Rumpf/1«, »Rechter_Flügel/1«, »Linker_Flügel/1«, »Leitwerk/1« und »Höhenruder/1«. So können Sie die Körper beider Flugzeuge problemlos unterscheiden.

3.6 Die Darstellungsarten (»Plotten«-Menü)

Die Darstellungsarten des »Animator« entsprechen weitgehend denen von »Construct«. In diesem Handbuch werden daher nur die Unterschiede behandelt. Alles weitere kann im Handbuch zu »Reflections« nachgelesen werden. Alle Funktionen befinden sich im »Plotten«-Menü, können aber auch über einen Tastendruck erreicht werden. Falls eine Funktion auch per Tastatur erreichbar ist, steht die entsprechende Taste in Klammern hinter dem Namen der Funktion. Die Taste wird im Gegensatz zu »Construct« auch alleine (ohne gleichzeitiges Festhalten der rechten [A]-Taste) akzeptiert.

3.6.1 Die Funktion Plotkörper (k)

Im Gegensatz zur gleichnamigen »Construct«-Funktion kann der Anwender hier nicht nur einen einzigen Körper angeben, sondern eine ganze Liste von Körpern, die dargestellt werden sollen. Dadurch läßt sich die Funktion relativ flexibel auf bestimmte Situationen anpassen, bei denen Kombinationen von Körpern betrachtet werden. Wird »Plotkörper« gewählt, so liefert der »Animator« im Auswahlfenster zunächst die aktuelle Plotkörperliste (anfangs nur ein Körper: »all«). Nach einem Klick ins Auswahlfenster fragt der »Animator«, ob die Liste geändert werden soll. Wenn ja, genügt ein Klick auf den Namen des entsprechenden Körpers und dieser wird in die Liste aufgenommen. Dieser Vorgang wird so lange wiederholt, bis alle gewünschten Körper in der Liste enthalten sind.

3.6.2 Die »Hidd«-Funktionen

Diese Funktionen (im Menü unter den Namen »Hidd0«, »Hidd1« und »Hidd2« zu finden und mit den Tasten 0, 1 und 2 erreichbar) sind identisch zu den entsprechenden in »Construct«.

3.6.3 Gitter

Hiermit läßt sich die Gitterfunktion ein- und ausschalten. Bei eingeschalteter Funktion wird bei jedem Plotten ein Raster über den Plotbereich gelegt. Dies ist besonders in der Parallel-Darstellung nützlich, um Körper exakt zu bewegen.

3.6.4 Wire-Frame (w)

Diese Funktion ist zusätzlich hinzugekommen, weil im »Animator« die Darstellung normalerweise auf die Skelette reduziert ist. Diese sind wesentlich weniger detailliert als die von »Construct« gewohnte Wire-Frame-Darstellung, dafür kostet der Bildschirmaufbau weniger Zeit. Um die Wire-Frame-Darstellung auch im »Animator« zu aktivieren, genügt die Wahl dieses Menüs oder ein Druck auf die Taste »w«. Zur Kontrolle besonders wichtiger Schlüsselpositionen kann dies mitunter empfehlenswert sein.

3.6.5 Sonstige

Alle weiteren Menüpunkte sind in ihrer Funktion identisch mit den entsprechenden aus »Construct« bereits bekannten und werden daher nicht näher erläutert. Die folgende Liste enthält alle Namen mit den entsprechenden Tastaturabkürzungen:

vorne (v)

hinten (h)

rechts (r)

links (l)

unten (u)

oben (o)

persp (p)

Bild*2 (5)

Außer den Funktionen, die per Menü erreichbar sind, gibt es auch einige, die über Gadgets im Plotfenster aktiviert werden. Sie dienen der bequemen Skalierung des Darstellungsbereiches im Plotfenster.

3.6.6 Zoom-Gadget

Links vom Plotbereich ist ein Proportionalregler, der die Vergrößerung des dargestellten Bildes stufenlos regelt (ausgehend von der aktuellen Vergrößerungsstufe). Die Vergrößerung kann im Bereich 0.2 (Verkleinerung um den Faktor 5) bis 5.0 (Vergrößerung um den Faktor 5) liegen. Möglicherweise ist es anfangs verwirrend, daß bei der Vergrößerung die Körper kleiner werden, aber der Begriff »Vergrößerung« bezieht sich hier auf den Plotbereich – und wenn ein größerer Ausschnitt betrachtet wird, schrumpfen die darin enthaltenen Körper scheinbar.

>,<,^,v (Statt dessen bitte eine Hardcopy dieser Gadgets einfügen.)

Mit diesen vier Gadgets kann der Plotbereich nach rechts, links, oben und unten bewegt werden. Mit jedem Anklicken verschiebt sich der Ausschnitt jeweils um $^{1}/_{10}$ des Plotfensters. Die Objekte bewegen sich dann scheinbar in die entgegengesetzte Richtung.

Die Gadgets an der linken Seite sind nur bei paralleler Darstellung (vorne, hinten, unten...) benutzbar. Die jetzt folgenden beziehen sich auf die perspektivische Darstellung. Sie sind weitgehend identisch zu den entsprechenden Gadgets von »Construct«. Sie werden aktiviert, sobald per Menüfunktion »persp« (oder Taste p) auf die perspektivische Darstellung umgeschaltet wurde.

RLD
Die Kamera dreht sich nach rechts bzw. links je nach Stellung des Schiebereglers.

RLB
Wie RLD, nur bewegt sich die Kamera anstatt zu drehen.

Tele
Teleobjektiv. Hier wird die Vergrößerung eingestellt.

Roll
Einstellung des Rollwinkels der Kamera.

VZB
Vor- und Zurück-Bewegen der Kamera (in Relation zur Blickrichtung).

HRD
Hoch- und Runter-Drehen.

HRB
Hoch- und Runter-Bewegen.

Die folgenden beiden Gadgets entsprechen der Funktion Kamera von »Construct«. Dort dreht sich die Kamera bei den Funktionen um den Mittelpunkt des zuvor ausgewählten Körpers. Hier dreht sie sich hingegen um den Mittelpunkt der Körper der Plotkörper-Liste.

Br
Einstellung des Breitengrades, auf dem sich die Kamera auf der Oberfläche einer fiktiven Kugel um den Mittelpunkt der Plotkörper-Liste dreht.

Lä
Entsprechend »Br«, nur handelt es sich hier um den Längengrad.

Bei Anklicken der beiden letzten Gadgets wird die aktuelle Kameraposition verlassen. Die Kamera wird dann in eine gedachte Kugel, um den Mittelpunkt der Plotkörper-

Liste, auf den entsprechenden Breiten- und Längengrad bewegt. Die anderen Gadgets gehen dagegen von der aktuellen Kameraposition aus und bewegen relativ dazu.

Es kommt durchaus vor, daß man die Plotkörper-Liste während der Kamera-Definition verändert. In diesem Fall kann es vorkommen, daß die Kamera bei Betätigung des »Br«- oder »Lä«-Gadgets sich eventuell völlig anders verhält als vorgesehen, da sich der Mittelpunkt der Plotkörper-Liste inzwischen verändert hat.

3|7| Das »Körper«-Menü

Das »Körper«-Menü enthält Funktionen, um die Manipulation von Körpern zu vereinfachen. In Klammern befinden sich auch hier die Tastatur-Abkürzungen. Die Menüpunkte im Überblick:

3|7|1| Ausrichten (a)

Normalerweise arbeitet der »Animator« im Weltkoordinatensystem. Oft ist es aber sinnvoll, statt dieser absoluten Bezugspunkte die Szene relativ zu einem Körper zu betrachten. Das ermöglicht die Funktion »Ausrichten«. Sie richtet alle Körper der aktuellen Szene nach einem beliebigen Körper aus. Die Funktion schaltet dabei automatisch in die achsenparallele Darstellung um. Das erleichtert manche Positionier-Vorgänge, ist jedoch von der Darstellungsweise her zunächst gewöhnungsbedürftig.

In welchem Koordinatensystem der »Animator« gerade arbeitet, ist an verschiedenen Dingen erkennbar: Ist das Weltkoordinatensystem aktiv, dann sieht das Plotfenster aus wie in »Construct« – mit den vier Himmelsrichtungen und den Positionen »oben« und »unten«. Ist dagegen das Koordinatensystem eines Körpers gültig, so sind diese Marken verschwunden, da sie nur im Weltkoordinatensystem gelten. Im Balken des Plotfensters ist zudem der Name des Körpers sichtbar, auf dessen Koordinatensystem »Animator« ausgerichtet ist.

Nach dem Wählen von »ausrichten« fragt der »Animator«, nach welchem Körper er ausrichten soll. Wird an dieser Stelle nichts angegeben, so behält der »Animator« die aktuelle Ausrichtung bei. Wird »all« gewählt, so werden alle Körper wieder auf das Weltkoordinatensystem ausgerichtet. Bei der Wahl eines anderen Körpers, beziehen sich alle Koordinaten danach in ihrer Ausrichtung auf diesen.

3|7|2| Eigenschaften

Diese Funktion erhält erst im Zusammenhang mit Gelenken eine Bedeutung und wird daher erst dort besprochen (Kapitel 3.15).

3|7|3| clear_eigenschaften

Auch für diese Funktion gilt das bei »Eigenschaften« Gesagte.

3.7.4 Körpernamen (n)

Viele Funktionen des »Animator« benötigen die Eingabe eines Körpernamens. Bei einer größeren Anzahl hat man jedoch schnell die Übersicht verloren. Dann ist es unerläßlich, sich die Namen anzeigen zu lassen. Diese Funktion ordnet jedem Körper seinen Namen zu.

3.7.5 Gelenknamen

Hiermit werden die Namen aller aktuellen Gelenke ins Plotfenster geschrieben. Denn bei Gelenken kann das gleiche Problem entstehen wie bei den Körpernamen. Es werden alle Gelenke gezeichnet, die zur aktuellen Sequenz gehören.

3.7.6 Zentrieren (z)

Diese Funktion sorgt dafür, daß das Zentrum aller Körper der Plotkörperliste im Mittelpunkt des Plotfensters liegt. Sie arbeitet sowohl im Parallel- als auch im Perspektive-Modus.

3.7.7 Lichtfarbe

Der »Animator« läßt die Veränderung der Lichtquelle innerhalb einer Sequenz zu. Dadurch werden Animationen möglich, deren Beleuchtung sich während des Filmablaufs verändert. Mit dem Menüpunkt »Lichtfarbe« wird die Helligkeit und Farbe der Lichtquellen verändert.

3.8 Veränderungen an Körpern

Es ist Sinn einer Computeranimation, den Eindruck, den ein Körper beim Betrachter erweckt, permanent zu verändern. Da dies der hauptsächliche Sinn des »Animator« ist, wurden die dazu vorgesehenen Funktionen – statt wie bei »Construct« in Menüs – in Gadgets eingebaut, wodurch sie ständig im Plotfenster sichtbar sind. Ansonsten sind die Funktionen im wesentlichen die gleichen geblieben, wie von »Construct« her bekannt.

Eine weitere Änderung ist die Art der Auswahl eines Körpers, um ihn zu verändern. Wurde dies bei »Construct« noch im Auswahlfenster erledigt, so geschieht das beim »Animator« nur noch in Ausnahmefällen. Statt dessen wählt der Anwender den Körper mit dem Mauszeiger aus. Ist ein Körper gewählt, so beziehen sich alle folgenden Operationen auf ihn. Er ist der sogenannte Arbeitskörper.

3.8.1 Der Arbeitskörper »Work«

Das Festlegen des Arbeitskörpers geschieht beim »Animator« auf eine ungewöhnliche Weise. Nachdem das Prinzip durchschaut ist, läßt sich damit allerdings extrem schnell und einfach arbeiten. Um sich mit der Funktionsweise vertraut zu machen, starten Sie

den »Animator« mit der Szene »robby«, die auf der Programmdiskette enthalten ist. Bewegen Sie den Mauszeiger in den Darstellungsbereich des Plotfensters, drücken Sie die linke Maustaste und halten Sie sie gedrückt. Bewegen Sie den Mauszeiger hin und her. Im Statusfenster erscheint jeweils der Name des Körpers, der am nähesten am Mauszeiger liegt. Zusätzlich wird der Körper mit gepunkteten Linien gezeichnet. Sobald Sie den gewünschten Körper angezeigt haben, lassen Sie einfach die linke Taste los. Damit ist der Körper gewählt, der in diesem Moment im Statusfenster angezeigt war (und der am nähesten beim Mauszeiger war).

Auf diesen Körper beziehen sich von jetzt an sämtliche Veränderungen, die der Anwender vornimmt. Es ist der Arbeitskörper, genannt »Work«. Sollte einmal kein Körper als derjenige mit dem kürzesten Abstand zum Mauszeiger definierbar sein, so wird im Statusfenster kein Name angezeigt und beim Loslassen der Maustaste fragt »Reflections-Animator« mit dem von »Construct« gewohnten Auswahlfenster, welcher Körper als »Work« definiert werden soll. Die Frage nach einer neuen Sequenz für den gewählten Körper kann in jedem Fall mit »nein« beantwortet werden, da eine Sequenz auch jederzeit später angefügt werden kann.

Zu beachten ist außerdem, daß nur Körper gewählt werden können, die in der Plotkörper-Liste enthalten sind, denn andere werden nicht im Fenster dargestellt. Soll ein anderer Körper bewegt oder verändert werden, so muß er erst in diese Liste aufgenommen werden.

Sobald ein Körper zum Verändern ausgewählt ist, füllt sich ein Teil der Gadgets am oberen Rand des Darstellungsbereichs mit Text. Solange kein Körper ausgewählt ist, haben auch die Gadgets keinen Sinn. Denn diese Felder sind für die Manipulationen am in der Statuszeile angezeigten Körper vorgesehen (siehe Bild 8). Die Gadgets sind in einer langen Reihe angeordnet nach Gruppen aufgeteilt.

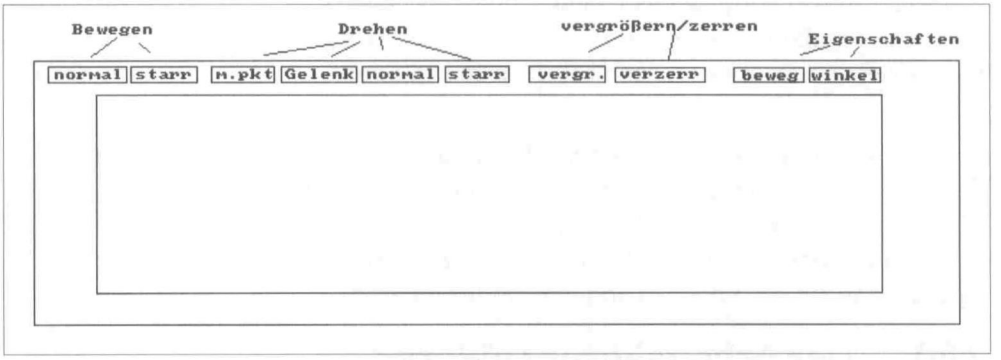


Bild 8: Gadgets für Manipulationen an Körpern

Die beiden Gadgets auf der linken Seite sind zum Bewegen des Körpers vorgesehen. In der Mitte sind vier zum Drehen, daneben zwei zum Vergrößern und Verzerren. Die beiden ganz rechts sind für das Einstellen von Eigenschaften des Körpers gedacht.

Die beiden Gadgets für das Bewegen von Körpern heißen »normal« und »starr«. Nach einem Klick auf das linke Gadget (»normal«) wechselt dessen Schriftfarbe von weiß auf orange. Damit ist ein neuer Bewegungsmodus aktiviert, aber sonst passiert noch nichts. Erst wenn sich der Mauszeiger über dem Darstellungsbereich befindet und die rechte Maustaste dort gedrückt wurde, läßt sich wieder eine Veränderung feststellen. Das Skelett des gewählten Körpers folgt jetzt jeder Mausbewegung, so lange, bis die rechte Maustaste nochmals gedrückt wird – an dieser Stelle wird der Körper eingerastet. Mit einem Druck auf die Taste [w] läßt sich schnell kontrollieren, ob die Position korrekt ist: Der Körper wird dann als Drahtgittermodell (w = »Wire-Frame«) dargestellt.

Die Funktionsweise der Gadgets ist immer wieder die gleiche: Sie werden durch Anklicken mit der linken Maustaste aktiviert (es sei denn, sie sind gerade gesperrt). Wird ein neues Gadget angeklickt, so ist automatisch die Aktivierung des letzten Gadgets gelöscht. Die Funktion der Gadgets im einzelnen (in der Reihenfolge von rechts nach links):

»normal«
Dabei handelt es sich um die normale Bewegungsfunktion wie von »Construct« her bekannt. Im Unterschied zu »Construct« wird hier aber nicht ein Rahmen bewegt, sondern der Körper selbst, beziehungsweise seine Skelettdarstellung. Nach dem ersten Klick mit der rechten Maustaste im Darstellungsbereich, folgt der »Work«-Körper der Mausbewegung bis zum nächsten Klick.

»starr«
Diese Funktion wird erst im Zusammenhang mit Gelenken interessant. Bei einer Sequenz mit durch Gelenke verbundenen Körpern enthält die Sequenz mehrere Körper gleichzeitig. Mit dem »starr«-Gadget werden sie alle gleichzeitig und unverändert in ihren Winkeln mitbewegt.

»m.pkt«
Dieses Gadget dreht den »Work«-Körper um seinen Mittelpunkt. Sobald die rechte Maustaste geklickt wird, dreht sich der »Work«-Körper analog zur Mausbewegung um seinen Mittelpunkt. Im Statusfenster wird derweil der aktuelle Drehwinkel angezeigt. Es wird so lange gedreht, bis erneut mit rechts geklickt wird.

»gelenk«
Diese Funktion ist erst im Zusammenhang mit Gelenken interessant. Sind keine Gelenke vorhanden, so bleibt sie gesperrt. Der »Work«-Körper kann hier um eines seiner Gelenke gedreht werden.

»normal«
Dieses Gadget hat den gleichen Effekt wie »m.pkt«, nur wird der »Work«-Körper anstatt um seinen Mittelpunkt, um den Punkt gedreht, an dem sich der Mauszeiger beim ersten Klick mit der rechten Maustaste befindet. Der zweite Klick startet die Drehung und mit dem dritten ist diese abgeschlossen.

»starr«
Dieses Gadget funktioniert ähnlich wie das Bewegungsgadget gleichen Namens: Alle durch Gelenke verbundenen Körper einer Sequenz drehen sich unverändert mit. Ansonsten funktioniert »starr« wie »m.pkt«.

»vergrößern«
Der »Animator« unterscheidet prinzipiell nicht zwischen Vergrößern und Verkleinern. Die Größenänderung wird aus dem Abstand des Mauszeigers vom Körpermittelpunkt errechnet. Nach dem ersten Mausklick wird eine Bewegung der Maus zum Mittelpunkt hin als Anweisung zum Verkleinern interpretiert, vom Mittelpunkt weg hingegen zum Vergrößern. Der Vergrößerungsfaktor wird ständig im Statusfenster angezeigt.

»verzerren«
Im Gegensatz zur Zerr-Funktion von »Construct« bezieht sich dieses Gadget stets auf die körpereigenen Koordinatensysteme. Ebenso wie bei »Construct« wird mit dem Mauszeiger in Richtung der x-, y- oder z-Achse verzerrt.

»beweg« und »winkel«
Diese beiden Gadgets am rechten Rand haben nur im Zusammenhang mit Gelenken eine Funktion, daher werden sie dort näher erläutert.

3.8.2 Verändern aus der Perspektivsicht

Eine weitere Neuerung gegenüber »Construct« ist die Möglichkeit, Körper aus der Perspektivsicht heraus zu bewegen. Diese Betrachtungsweise wird mit dem Menü »plotten.persp« (oder durch Drücken der p-Taste) eingeschaltet. Wird jetzt ein Körper bewegt, so verändert sich jetzt gleichzeitig seine Darstellung in perspektivisch korrekter Form. Dabei taucht allerdings ein Problem auf: In der Paralleldarstellung wurden immer die exakten Himmelsrichtungen des Weltkoordinatensystems angezeigt. Dadurch ist stets sofort klar, in welche Richtung die Maus den Körper verschiebt. In der Perspektivdarstellung dagegen wird die Szene aus Sicht der Kamera gezeigt. Dadurch gehen die exakten Richtungsinformationen verloren. Aber der »Animator« merkt sich

die letzte Einstellung in der Paralleldarstellung und richtet sich mit allen Veränderungen nach dieser letzten Einstellung. Soll die Richtung verändert werden, so geht dies nur mit einem Wechsel in die Paralleldarstellung.

3.9 Eine kurze Einführung in Sequenzen

Das letzte Kapitel behandelte das Bewegen und Drehen von Körpern. Für sich genommen ist beides nicht gerade weltbewegend, denn das geht schließlich auch mit »Construct«. Für eine Animation ist ein weiterer Faktor wichtig: die Zeit. Damit aus Bildern ein Film entsteht, genügt schließlich nicht allein die Beschreibung der Darsteller und Kulissen, sondern im Drehbuch muß auch verzeichnet sein, wann welches Element an welchem Platz und aus welcher Perspektive sichtbar ist. Diese Beschreibung nennt man bei Animationen einen Bewegungspfad oder eine Sequenz.

Der »Reflections-Animator« ist ein »Keyframe«-System – das bedeutet, daß der Regisseur Körperpositionen in einigen Schlüsselbildern vorgibt, alle dazwischenliegenden Bilder werden vom Programm selbständig berechnet.

Dabei ist der Begriff »Keyframe« etwas irreführend, denn der »Animator« arbeitet eigentlich nicht mit der Beschreibung von Bildern (oder »Frames«). Statt dessen erlaubt das Programm dem Regisseur, den logischeren Weg zu gehen, indem er für jeden Körper festlegt, zu welchem Zeitpunkt er an welchem Ort sein soll – ohne sich Gedanken darüber zu machen, das wievielte Bild dies gerade ist. Formal gesprochen sähe das folgendermaßen aus:

»Zum Zeitpunkt t1 ist Körper »A« an der Position x1. Zum Zeitpunkt t2 ist Körper »A« an Position x2.« Die Zwischenpositionen im Zeitraum zwischen t1 und t2 kann stufenlos interpoliert werden.

Die Alternative wäre, dem Programm mitzuteilen, »das Bild Nummer n besteht aus Objekt x1 an der Position y1, Objekt x2 an der Position y2, Objekt x3 an der Position y3 usw. und in Bild Nummer n+z sind die Positionen y1+z, y2+z, y3+z usw.« – zweifellos etwas umständlicher.

In der Praxis ist die Vorgehensweise relativ einfach: Der Anwender bewegt den »Work«-Körper, so wie im letzten Kapitel erläutert, auf die gewünschte Position. Im Sequenzfenster rechts oben ist die aktuelle Sequenz angezeigt, das ist die, zu der der »Work«-Körper gehört. Am Anfang gibt es noch keine Sequenz.

Ist der Körper auf der gewünschten Position, gibt der Anwender im Sequenzfenster mit dem Zeitcursor einen Zeitpunkt an, zu dem der Körper seine Position erreichen soll. Ein Klick auf eines der Gadgets übernimmt die augenblickliche Position in die Sequenz. Welche Bedeutung die einzelnen Gadgets haben, wird im nächsten Kapitel behandelt.

Der Vorgang läßt sich beliebig oft wiederholen. Sind mehrere Schlüsselpositionen gespeichert, dann ist es ganz einfach, eine Zwischenposition des Körpers zu ermitteln: Es muß nur der Zeitcursor im Sequenzfenster auf den Zeitpunkt gesetzt werden, für den die Zwischenposition errechnet werden soll. Nach dem Klick auf ein Gadget erscheint die Zwischenposition im Plot-Fenster.

Der »Animator« speichert nicht für jede Position sämtliche Daten eines Körpers, das wäre viel zu speicheraufwendig und hätte auch noch den Nachteil, daß es bei Berechnung der Zwischenposition zur Verformung des Körpers kommen kann. Vielmehr werden nur wenige Parameter festgehalten, die aber ausreichen, die Position des Körpers aus seiner Grundposition zu rekonstruieren.

Die Parameter sind folgende:

- Die Position des Körpermittelpunktes (3 Koordinaten).
- Die Orientierung des Körpers (3 Winkel).
- Die Größe des Körpers.
- Einige Attribute.

Dadurch können schon eine Menge von Schlüsselpositionen eingegeben werden, ohne gleich den ganzen Speicher zu füllen. Entsprechend komplexe Bewegungsabläufe sind auch mit geringem Speicher möglich.

Einmal definierte Bewegungsabläufe lassen sich auch nachträglich auf mehrere Arten ändern:

- Schlüsselpositionen können wieder gelöscht werden.
- Schlüsselpositionen können auf der Zeitachse stufenlos verschoben werden, was einem Beschleunigungs- oder Bremseffekt entspricht.
- Neue Schlüsselpositionen können eingefügt werden, um den Bewegungsablauf zu verfeinern.
- Schlüsselpositionen lassen sich verdoppeln, um bestimmte Bewegungen zu wiederholen.
- Schlüsselpositionen können von anderen Sequenzen in gewissem Umfang übernommen werden.

Jede Sequenz hat eine genau definierte Gültigkeitsdauer. Sie reicht vom Zeitpunkt der ersten Schlüsselposition bis zum Zeitpunkt der letzten.

Ein Körper kann auch zu mehreren Sequenzen gehören, oder genauer: Es kann mehrere Sequenzen geben, die einen Bewegungsablauf für den gleichen Körper enthalten. Dies ist beispielsweise dann sinnvoll, wenn eine Sequenz zu einem Zeitpunkt beginnt, wo eine andere gerade aufhört.

Körper können genau dann mit Gelenken verbunden werden, wenn die beteiligten Körper Brüder (siehe Kapitel 3.4) sind. Sie sind dann aneinander gekoppelt, was sich beim Bewegen und Drehen auswirkt. Gelenke können nur beim Erzeugen einer Sequenz definiert werden. Sie gehören zur Sequenz, nicht zu den Körpern, das ist für das Verständnis der Funktionsweise sehr wichtig.

Wenn eine neue Sequenz erzeugt wird, fragt der »Animator« daher immer, ob der »Work«-Körper mit anderen Körpern verbunden werden soll (falls er Brüder hat). Sie können im Auswahlfenster dann die Körper auswählen, mit denen »Work« verbunden werden soll. Anschließend werden vom Benutzer die Gelenke definiert.

Gelenke werden ab dem Kapitel 3.15 genauer erläutert.

3.10 Erzeugung einfacher Sequenzen

Um sich mit der Verwendung von Sequenzen vertraut zu machen, ist es sinnvoll, das Folgende Schritt für Schritt nachzuvollziehen.

Starten Sie dazu den »Animator« mit der Testszene »test« von der Programmdiskette. Wählen Sie einen der Buchstaben als »Work«-Körper, beispielsweise das »T«. Auf die Frage, ob für »T« eine neue Sequenz eröffnet werden soll, antworten Sie diesmal mit ja. Den Vorschlag für den Namen (»T«) können Sie getrost übernehmen. Der »Animator« fragt darauf, mit welchen Körpern »T« verbunden werden soll (er schlägt drei vor: »E«, »T1« und »S«). Das Verbinden ist jedoch nur bei Gelenken sinnvoll. Wählen Sie daher keinen Körper an, sondern brechen Sie entweder durch das Drücken der rechten Maustaste ab oder klicken Sie einfach in ein leeres Feld.

Damit haben Sie eine Sequenz für den Körper »T« erzeugt. Im Sequenzfenster erscheint eine Zeitskala für »T«. Ein Klick mit der linken Maustaste im weißen Feld setzt den Zeitcursor (das ist das kleine Dreieck mit dem senkrechten Strich) an die Stelle des Mausklicks. Sobald das Sequenzfenster aktiv ist, scrollt die Zeitskala jeweils in Richtung der Mausbewegung, sobald der Mauszeiger an den linken oder rechten Rand des Fensters kommt. Außerdem läßt sich mit dem »<>«-Gadget den Darstellungsbereich der Zeitskala vergrößern und mit »><« verkleinern. Rechts neben der Zeitskala ist ein Feld mit einigen Gadgets. Sie werden weiter unten noch ausführlich erklärt. Das Gadget mit der Aufschrift »Snap« ist jedoch jetzt schon interessant. Klicken Sie es einmal an. Darauf erscheint unterhalb des Zeitcursors eine rote Linie und der Cursor wandert ein Stück weiter. Damit ist bereits eine Schlüsselposition samt Zeitpunkt festgelegt. Sobald Sie »Snap« anklicken, wird die augenblickliche Position und Orientierung des Körpers gespeichert zusammen mit dem Zeitpunkt, der durch den Zeitcursor gekennzeichnet ist.

Das Schema zum Erzeugen von Schlüsselpositionen ist somit folgendes:

- Bewegen/Drehen/Vergrößern des Körpers bis er die gewünschte Position erreicht hat.

- Kennzeichnen des richtigen Zeitpunkts mit dem Zeitcursor.
- »Snap« anklicken.

Es fällt noch auf, daß anfangs über der roten Linie ein blauer Strich erscheint. Dieser Strich kennzeichnet die erste und letzte Schlüsselposition der Sequenz, und damit gleichzeitig die Sequenzdauer. Anfangs ist die Dauer ja auf einen Punkt begrenzt. In Zukunft verwenden wir in diesem Handbuch statt des Begriffs »Schlüsselposition« den Begriff »Snap«.

Um die Sequenz zu vervollständigen (denn ein Film, dessen Darsteller sich die ganze Zeit nicht vom Fleck bewegen, wird schnell fad), bewegen Sie das »T« an einige neue Positionen und machen Sie dort in der soeben angewendeten Form noch weitere »Snaps«.

Um die Fähigkeiten des »Animator« beim Interpolieren von Positionen zu testen, machen Sie einmal folgenden Versuch: Fahren Sie mit dem Mauszeiger zu einem Zeitpunkt zwischen zwei »Snaps« und klicken Sie einmal kurz mit rechts. Kurz darauf erscheint das Skelett von »T« in der Position und Orientierung, wie es dem Zeitpunkt des Zeitcursors entspricht. Mit der rechten Maustaste lösen Sie die Interpolation des Körpers für den Zeitpunkt aus, an dem sich der Mauszeiger befindet. »Animator« errechnet die Position von »T« aus der Position der beiden benachbarten »Snaps« und der zugehörigen Zeitpunkte. Wiederholen Sie den ganzen Vorgang, und bewegen Sie zuvor den Zeitcursor etwas weiter. Es ist zu erkennen, daß »T« der zuvor durch die »Snaps« bestimmten Bahn folgt.

Damit können Sie kontrollieren, ob die Bahn des Körpers, wie sie vom »Animator« interpoliert wird, Ihren Wünschen entspricht. Sind Sie nicht zufrieden, können Sie jederzeit eine interpolierte Position des Körpers wie gewohnt verschieben, drehen oder vergrößern und an der Stelle einfach einen neuen »Snap« setzen. Das dürfte der einfachste Weg sein, um schnell das gewünschte Ergebnis zu erreichen:

- Körper an die Anfangsposition bewegen – »Snap«
- Körper an die Endposition bewegen – »Snap«
- Mit dem Mauszeiger an den zeitlichen Mittelpunkt zwischen beiden »Snaps« gehen – rechts klicken
- Wenn zufrieden mit der Position, dann mit »ok« beenden, sonst:
- Körper an gewünschte Position bewegen – »Snap«
- Schritte 3 bis 6 ausführen bis die Position exakt die gewünschte ist.

Dieses schrittweise Interpolieren kann aber nur eine Notlösung während der Konstruktionsphase sein. Bevor der Film berechnet wird, muß zumindest einmal ein halbwegs realistischer Eindruck des Bewegungsablaufs entstehen. Das erreicht der

»Animator« entweder sofort mit der Skelett-Animation oder innerhalb weniger Minuten mit der Wire-Frame-Animation. Beide werden in Kapitel 3.8 ausführlich erläutert.

3.11 Das Sequenz-Fenster

Die wesentlichen Funktionen zum gezielten Manipulieren von Sequenzen sind im Sequenz-Fenster (in der rechten oberen Ecke) zusammengefaßt. Im rechten Teil befindet sich ein Feld mit 18 Gadgets, während sich im linken Teil die Zeitskala zum Verändern der »Snaps« per Mausklick befindet.

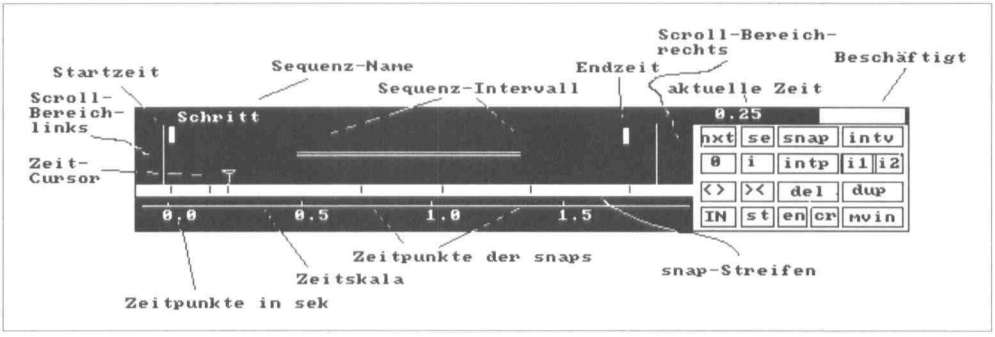


Bild 9: Das Sequenz-Fenster besteht aus zwei Teilen – der Zeitskala und dem Gadgetfeld

Bild 9 zeigt das Sequenz-Fenster. Betrachten wir zuerst einmal den linken Teil:

Zeitskala: Der waagerechte Balken ist die Zeitskala – die Markierungen darunter stehen für die einzelnen Zeitabschnitte, beispielsweise halbe Sekunden.

»Snaps«: Die »Snaps« sind durch die senkrechten Striche auf der Zeitskala dargestellt. Die »Snaps« befinden sich in dem schwarzen Streifen. Die Position der Striche über der Zeitskala bestimmt eindeutig den Zeitpunkt des Erreichens einer Position. Sollten zwei »Snaps« so nahe beieinander liegen, daß die entsprechenden »Snap«-Striche übereinander gezeichnet werden, gibt es einen blauen Strich statt zweier roter. Blauer »Snap«-Strich bedeutet also, daß an dieser Stelle mehr als ein »Snap« liegt.

Zeitcursor: Über den »Snaps« befindet sich der Zeitcursor (falls er im sichtbaren Bereich des Fensters liegt). Es ist der senkrechte Strich mit dem Dreieck darüber. Er zeigt den aktuellen Zeitpunkt der Sequenz an. Dieser Zeitpunkt ist zudem als Zahl über dem Gadget-Feld dargestellt. Die aktuelle Zeit hat mehrere Bedeutungen. Die wichtigste ist das Positionieren von »Snaps«. Der Zeitcursor kann einfach durch Bewegen des Mauszeigers und Klicken mit links bewegt werden.

Snap_Streifen: Die »Snaps« sind nicht auf dem weißen Hintergrund des Sequenzfensters dargestellt, sondern im schwarzen Snap_Streifen, und zwar aus

folgendem Grund: Es kommt oft vor, daß man einzelne »Snaps« auf der Zeitachse verschieben will, wenn man der Ansicht ist, eine bestimmte Schlüsselposition eines Körpers sollte früher oder später sein. Wenn Sie den Mauszeiger in den Snap_Streifen bewegen und die linke Maustaste drücken, sucht »Animator« den nächsten »Snap« und folgt daraufhin jeder Mausbewegung, bis die Taste wieder losgelassen wird. Ein »Snap« kann so frei zwischen seinen Nachbar-»Snaps« verschoben werden.

Intervall: Die über dem Cursor befindliche waagerechte Linie stellt ein bestimmtes Zeitintervall dar. Dieses Intervall ist für bestimmte Funktionen der Sequenz von Bedeutung. Aber auch zur Bestimmung des Zeitbereichs einer Animation. Die Linie ist rot und blau gezeichnet.

Start/End-zeit: Darüber sind wieder senkrechte Linien (in diesem Fall blau), die anzeigen, wo die Sequenz beginnt bzw. endet.

Beschäftigt: Oben rechts über dem Gadget-Feld ist ein rechteckiges Feld, das rot gefüllt wird, wenn eine Funktion für die Berechnung etwas länger braucht als »sofort«.

Scroll-Bereich-links: Der Scroll-Bereich liegt zwischen dem linken Rand des Sequenzfensters und der senkrechten schwarzen Linie daneben. Sobald der Mauszeiger in diesen Bereich kommt, scrollt die Zeitskala (und alle »Snaps«) nach rechts. Der sichtbare Ausschnitt verschiebt sich also auf der Zeitskala nach links.

Scroll-Bereich-rechts: Genau der umgekehrte Fall zum linken Bereich. Der sichtbare Ausschnitt bewegt sich zeitlich nach rechts.

3.1.1. Die Gadgets im Sequenzfenster

Um schnell interessante Effekte mit Sequenzen zu erreichen, sind die 18 Gadgets rechts neben der Zeitskala von entscheidender Bedeutung. Damit werden bisher ungeahnte Effekte möglich. Im folgenden werden die einzelnen Felder in der Reihenfolge von links oben nach rechts unten beschrieben. Zusätzlich sind sie über ihre Beschriftung identifizierbar (im Text fett gedruckt).

nxt: Bei einem Klick auf »nxt« bewegt sich der Zeitcursor automatisch auf die Position (bzw. den Zeitpunkt) des nächstgelegenen »Snaps«.

»Snap«: Diese Funktion wurde bereits kurz besprochen. Mit »Snap« wird ein Schnappschuß der augenblicklichen Position und Orientierung der Körper aus der Sequenz als Schlüsselszene festgelegt. Der Zeitpunkt des »Snaps« ist durch die Position des Zeitcursors vorgegeben. Bevor Sie »Snap« klicken, sollten Sie sich daher kurz vergewissern, ob der Zeitcursor auch an der richtigen Stelle ist. Nach dem »Snap« springt der Zeitcursor ein Stückchen weiter, um zu vermeiden, daß aus Versehen zwei Positionen auf dieselbe Stelle gesnapt werden.

se: Entspricht im wesentlichen der »Snap«-Funktion, nur wird hierbei zuvor der »Snap«, der sich am nächsten zum Zeitcursor befindet, gelöscht und dann ein neuer »Snap« an der gleichen Stelle gesetzt.

intv: »intv« steht für Intervall. Gemeint ist damit ein Zeitintervall, auf das sich mehrere Funktionen beziehen sollen, beispielsweise auch die Erzeugung der Animation. Benötigt werden dazu zwei Zeitpunkte auf der Skala. Der Einfachheit halber ist der erste Zeitpunkt bereits durch den Zeitcursor bestimmt. Über dessen Position erscheint dann auch ein kleiner Strich. Bei einem Klick auf »intv« nimmt der »Animator« automatisch den aktuellen Zeitpunkt (auf dem der Cursor steht) als den ersten Endpunkt des Intervalls und wartet dann, bis der Anwender mit der linken Maustaste im Sequenzfenster klickt. Dieser Zeitpunkt ist dann der zweite Endpunkt des Intervalls. Das Intervall wird dann durch die rot-blaue Intervall-Linie angezeigt. Wollen Sie den zweiten Zeitpunkt außerhalb des dargestellten Bereichs legen, muß nur der Mauszeiger rechts oder links in den Scroll-Bereich bewegt werden. Die Skala folgt dem Mauszeiger in die jeweilige Richtung.

Die nächste Reihe von Gadgets enthält ausschließlich Funktionen zum Interpolieren:

0: Hierbei werden alle Körper der Sequenz in ihre Null-Lage versetzt. Das ist normalerweise die Lage, die sie in der von »Construct« gelieferten Ursprungsdatei haben. Dabei wird aber die Gelenkstruktur beachtet, d.h., manche Körper werden eventuell noch ein wenig verschoben, damit sie an der richtigen Stelle bezüglich ihrer Gelenke sind. Diese Funktion ist nützlich, um wieder in eine definierte Ausgangslage zu kommen.

intp: »intp« hat die gleiche Wirkung wie ein Klick mit der rechten Maustaste auf die Zeitskala, also das Interpolieren von Zwischenpositionen. Interpolieren ist eine der zentralen Funktionen des »Animator«. Bei einem Klick auf »intp« berechnet der »Animator« die Position und Orientierung der Körper der Sequenz am aktuellen Zeitpunkt (der durch den Zeitcursor festgelegt ist). Um zu erfahren, wo sich ein Körper zu einem bestimmten Zeitpunkt befindet, muß nur der Zeitcursor an den betreffenden Punkt bewegt und dann »intp« geklickt werden. Liegt der Zeitcursor außerhalb des Sequenzbereiches, also entweder vor der Start- oder hinter der Endzeit, so hat »intp« keine Wirkung. Während des Arbeitens von »intp« erscheint die rote »Beschäftigt«-Marke. Bei Sequenzen mit vielen Körpern (besonders mit vielen Gelenken) kann dieser Vorgang einige Minuten dauern.

i: Die Wirkung von »i« ist im Prinzip die gleiche wie von »intp«, nur wird hierbei die augenblickliche Position der Körper nicht verändert, sondern nur die Orientierung interpoliert und an der augenblicklichen Position der Körper eingesetzt. Bei »intp« wird ja zusätzlich noch die Position interpoliert. Das macht die »i«-Funktion zu einem

leistungsfähigen Werkzeug, wenn es darum geht, die Orientierung eines Körpers in der augenblicklichen Position an einen anderen Zeitpunkt zu übernehmen.

i1: »i1« hat ebenfalls fast die gleiche Wirkung wie »intp«, mit dem Unterschied, daß vor dem Interpolieren das Plotfenster nicht gelöscht wird. Dies hat den Vorteil, daß die Verschiebung der Körper unmittelbar ersichtlich wird. Die mehrmalige Anwendung von »i1« an verschiedenen Punkten der Zeitskala hinterläßt eine Art Schleifspur des Bewegungspfads. Das kann für einen schnellen Überblick hilfreich sein, wenngleich mit der Zeit das Plotfenster vollgezeichnet wird.

i2: »i2« interpoliert die Position und Orientierung eines Körpers genau an der Stelle des »Snaps«, der am nähesten beim Zeitcursor liegt.

<>: Durch einen Klick auf »<>« wird der Darstellungsbereich verdoppelt (statt Halbsekundenschritten wird die Skala in Sekundenschritten fortgezählt). Man bekommt so einen größeren Überblick über die »Snaps« unter Verlust von Genauigkeit der Zeitskala.

><: Die Umkehrfunktion zu »<>«. Der Darstellungsbereich wird halbiert. Man kann sich so auf einen Bereich genauer einstellen.

del: Mit »del« (für »delete«) kann ein »Snap« gelöscht werden. Dabei wird jeweils der dem Zeitcursor am nähesten gelegene »Snap« gelöscht. Das bewirkt unter Umständen eine massive Änderung des Bewegungspfades.

dup: Um eine bestimmte Bewegung zu wiederholen, dupliziert man einfach die »Snaps« der Bewegung mit »dup«. Ein Beispiel wäre etwa ein Schritt. Mit »dup« werden daraus zwei Schritte und durch mehrfaches Duplizieren ein Gang. Um dem »Animator« mitzuteilen, welche »Snaps« dupliziert werden sollen, legt man einfach das Sequenz-Intervall (mit »intv«) über die entsprechenden »Snaps«. Überzeugen Sie sich zuvor (mit »intp«), ob es sich auch um die richtigen »Snaps« handelt. Sodann werden durch Klicken auf »dup« alle »Snaps«, die innerhalb des Intervalls liegen, verdoppelt und an die Sequenz angefügt.

HINWEIS: Die »Snaps« werden dabei so verändert, daß die Position des ersten duplizierten »Snaps« mit der des bisherigen letzten »Snaps« überstimmt. Das ist nützlich, wenn Schritte definiert sind und dupliziert werden sollen. Sie reihen sich dann (fast) nahtlos aneinander. Gelegentlich ist noch etwas Handarbeit zur Feinabstimmung nötig.

st: »st« steht für Startzeit. Man kann mit »st« den Zeitcursor auf die Startzeit einer anderen Sequenz einstellen. Die Wahl der betreffenden Sequenz erfolgt im Auswahlfenster. Der Sinn liegt in Sequenzen, die unmittelbar aufeinander folgen. Wenn die aktuelle Sequenz beispielsweise exakt an der Stelle enden soll, wo eine andere beginnt, so setzen Sie hier einfach den Zeitcursor auf die Startzeit der anderen Sequenz und setzen dann den letzten »Snap«.

en: Die gleiche Funktion wie »st«, nur wird hierbei der Zeitcursor auf die Endzeit einer anderen Sequenz gesetzt.

cr: Mit »cr« können Sie die »Snaps«, die innerhalb eines Intervalls (intv) liegen, zeitlich zusammendrücken. Wie sie zusammengedrückt werden, hängt davon ab, ob der Zeitcursor von links oder rechts kommt (siehe Bild 10). Bei Wahl von »cr« wird der Anwender aufgefordert, im Sequenzfenster die Startposition des Zeitcursors anzuklicken. Damit kann er festlegen, ob er nach links oder rechts zusammendrücken will. Danach genügen einige Mausbewegungen, um die »Snaps« zu justieren.

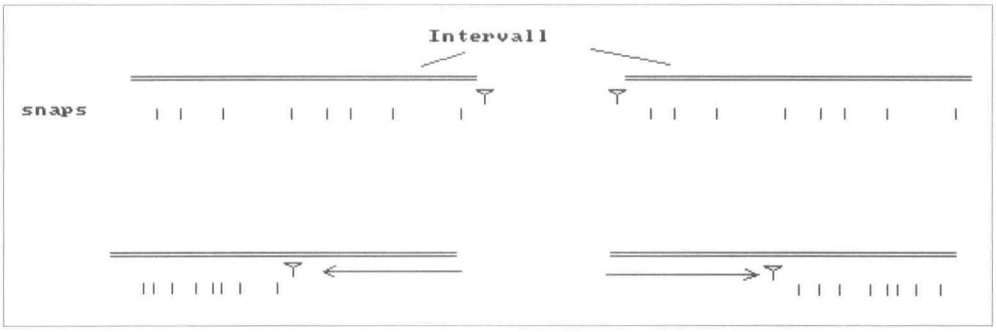


Bild 10: Die Lage des Zeitcursors beeinflußt die Art, wie ein Intervall gestaucht wird

mvin: Das letzte Gadget ist eine Funktion, um alle »Snaps« innerhalb des Intervalls mit konstanten Abständen zu verschieben. Nach Wahl von »mvin« verschieben sich alle »Snaps« des Intervalls analog zur Mausbewegung bis zum nächsten Klick.

HINWEIS: Bei »mvin« und »cr« sollte man beachten, daß »Snaps« nur verschoben oder gedrückt werden können, bis sie an der einen oder anderen Seite mit ihrem Nachbar-»Snap« zusammenstoßen.

3.12 Das Sequenzmenü

Die wichtigsten Funktionen zum Beeinflussen von Sequenzen liegen sicherlich in den 18 Gadgets, die bereits im letzten Kapitel beschrieben sind. Dennoch bleiben einige seltener gebrauchte Funktionen übrig, die daher in die Menüleiste aufgenommen wurden. Sie werden in diesem Kapitel behandelt. Die einzelnen Menüpunkte:

akt_sequ: Hier kann man eine neue Sequenz als die aktuelle bestimmen, wenn man bereits mehrere definiert hat. Man wählt sie per Namen im Auswahlfenster. Dies funktioniert auf die gleiche Weise wie beim Wählen eines Körpers als »Work«-Körper (siehe Kapitel 3.8). Beim Wählen eines Körpers, zu dem bereits eine Sequenz existiert, wird automatisch dessen Sequenz zur aktuellen.

neu_seq: Mit »neu_seq« kann eine neue Sequenz eröffnet werden. Die gleiche Wirkung erreicht man bei Körpern, indem ein Körper als »Work«-Körper ausgewählt wird, zu dem noch keine Sequenz existiert. Bei »neu_seq« erscheinen drei Untermenüpunkte, mit denen der Anwender bestimmt, welche Art Sequenz erzeugt werden soll. Der Normalfall ist eine Körpersequenz. Sie enthält die Bewegungsdaten eines (oder mehrerer) Körpers. Daneben gibt es noch Kamera- und Lichtsequenzen.

Körper: Wird der Untermenüpunkt »Körper« gewählt, so fragt der »Animator« zunächst, zu welchem Körper die Sequenz gehören soll. Im String-Fenster gibt der Anwender einen Namen für die Sequenz an. Das Programm schlägt den Körpernamen vor. Die nächste Frage (nach den Verbindungen) kann abgebrochen werden, es sei denn, der Körper soll Gelenke enthalten.

Kamera: Hiermit wird eine Kamera-Sequenz geöffnet. Bei einer Animation braucht man sich nicht auf die Bewegung von Körpern beschränken, sondern kann auch Kamerafahrten erzeugen. Hierzu muß nur ein Name für die Sequenz angegeben werden. Das Arbeiten mit Kamera-Sequenzen funktioniert ebenso wie mit Körper-Sequenzen. Wenn die parallele Darstellung eingeschaltet ist, kann die Kamera wie jeder Körper per Maus bewegt werden (die Kamera ist ein weißes Kreuz auf dem Bildschirm). Bei perspektivischer Darstellung erfolgt die Kamerabewegung und -drehung über die Kamera-Gadgets.

Licht: Genau wie einen Körper oder die Kamera kann auch eine Lichtquelle bewegt werden. Der Anwender gibt lediglich an, welche der maximal fünf Lichtquellen bewegt werden soll. Bei Lichtquellen kann man zusätzlich zur Bewegung auch noch die Farbe bestimmen (im Körper-Menü) und im Laufe des Films ändern. Zu Licht-Sequenzen kommen weiter unten noch mehr Informationen.

Take_Snaps: Oftmals ist es wünschenswert, die Bahn einer anderen Sequenz zu übernehmen und dann zu verändern. Diese Aufgabe übernimmt »Take_Snaps«. Diese Funktion ist relativ einfach zu bedienen. Das Programm fragt, von welcher Sequenz die »Snaps« übernommen werden sollen und zeigt alle Möglichkeiten an. Nach der Auswahl per Mausklick werden alle »Snaps« der gewählten Sequenz bei der aktuellen Sequenz angefügt.

HINWEIS: Man kann natürlich nicht die »Snaps« und damit die Bewegungsabläufe beliebiger Sequenzen übernehmen. Es gibt daher einige Einschränkungen, die sich danach richten, welche Körper sich in beiden Sequenzen (der aktuellen und der zu übernehmenden) befinden. Man unterscheidet folgende Fälle:

- Die aktuelle Sequenz hat einen Körper: Dieser Fall ist problemlos. Hat die andere Sequenz auch einen Körper, so werden ihre »Snaps« übernommen. Hat sie mehrere, so fragt das Programm, von welchem Körper die »Snaps« übernommen werden sollen.

- Die aktuelle Sequenz hat mehrere Körper: Dies geht jetzt nur, wenn alle Körper der aktuellen Sequenz auch in der anderen Sequenz vorkommen oder alle Körper der aktuellen Sequenz zumindest als »Doppelgänger« in der anderen Sequenz enthalten sind (zu Doppelgängern siehe Kapitel 3.5 »Einlesen von »Construct«-Szenen«). Der »Animator« pickt sich dann automatisch die »Snaps« der richtigen Körper heraus.

Gelenke: Diesem Menüpunkt ist ein eigenes Kapitel (3.15) gewidmet, er wird daher an dieser Stelle vernachlässigt.

Flags: Mit »Flags« kann man für die aktuelle Sequenz einige interessante Eigenschaften für Sequenzen einstellen. Sie werden über Untermenüpunkte ausgewählt. Die jeweils eingestellte Eigenschaft ist durch ein Häkchen gekennzeichnet.

Die ersten drei »Flags« bestimmen die Art, wie eine Kurve aus den »Snaps« berechnet wird.

hüpf: »Hüpf« ist die einfachste Variante. Dabei bleibt die Position eines Objekts zeitlich so lange erhalten, bis der nächste »Snap« kommt. Es wird nicht interpoliert. Der Körper hüpft statt dessen von »Snap« zu »Snap«. Diese Spezialform einer Sequenz dürfte vermutlich weniger oft vorkommen, ist aber beispielsweise für simulierte Schnitt-Effekte interessant. Oder Körper verschwinden lassen, die an anderer Stelle wieder auftauchen.

linear: Bei diesem »Flag« wird die Bahnkurve linear aus den »Snaps« interpoliert. Ein Körper bewegt sich damit auf Geraden zwischen den »Snaps«.

weich: Dieses »Flag« gibt der Bewegung einen sehr natürlichen Eindruck. Die Bahnkurve wird nämlich zu einer wirklichen Kurve, die sich weich von »Snap« zu »Snap« bewegt. Die »Snaps« werden dabei als Stützstellen einer sogenannten Bezier-Spline-Kurve betrachtet. Besonders für fliegende Körper und die Kamera ist dieses »Flag« zu empfehlen. Allerdings sollten Sie sich bei der weichen Interpolation die Bahnkurve anschauen (siehe unten). Manchmal werden dabei nämlich Unregelmäßigkeiten in die Kurve hineininterpoliert. Um auch noch so wild angeordnete »Snaps« mit einer weichen Kurve zu verbinden, muß eben manchmal etwas ausgeholt werden. Wo das nicht paßt, müssen Sie mit weiteren »Snaps« die Kurve auf die von Ihnen gewünschte Bahn »biegen«.

Die Unterschiede dieser »Flags« sind in Bild 11 verdeutlicht.

Die nächsten beiden »Flags« ermöglichen die Änderung der Orientierungsrichtung eines Körpers abweichend zur im »Snap« festgelegten Orientierung. Wenn Sie eines von den beiden »Flags« setzen, sollten Sie sich aber im klaren über die Orientierung des Körpers im Grundzustand sein. Beachten Sie dazu die Hinweise im Kapitel 3.5 Einlesen von »Construct«-Szenen.

Richtung: Das Richtungsflag veranlaßt den »Animator«, beim Interpolieren den Körper so zu drehen, daß er genau in Richtung seiner Bahnkurve orientiert ist. Dieses »Flag« ist besonders im Zusammenhang mit dem Weichflag bei fliegenden Körpern interessant.

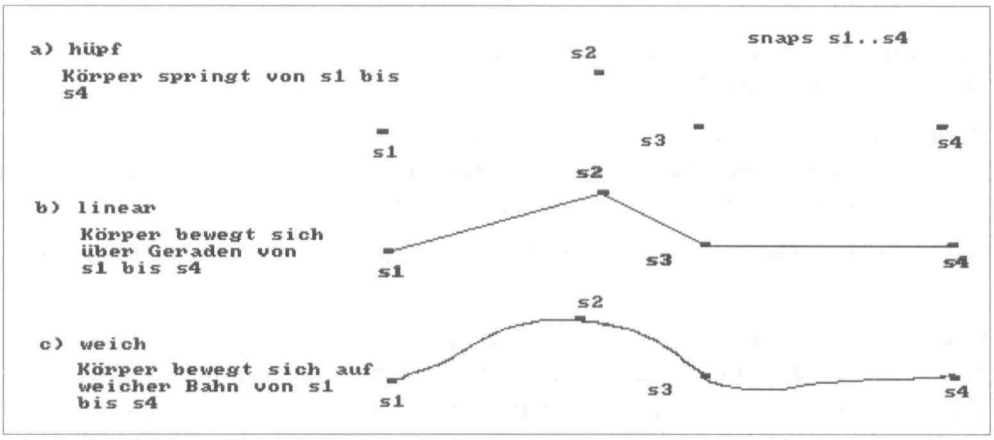


Bild 11: Flags beeinflussen die Art der Interpolation

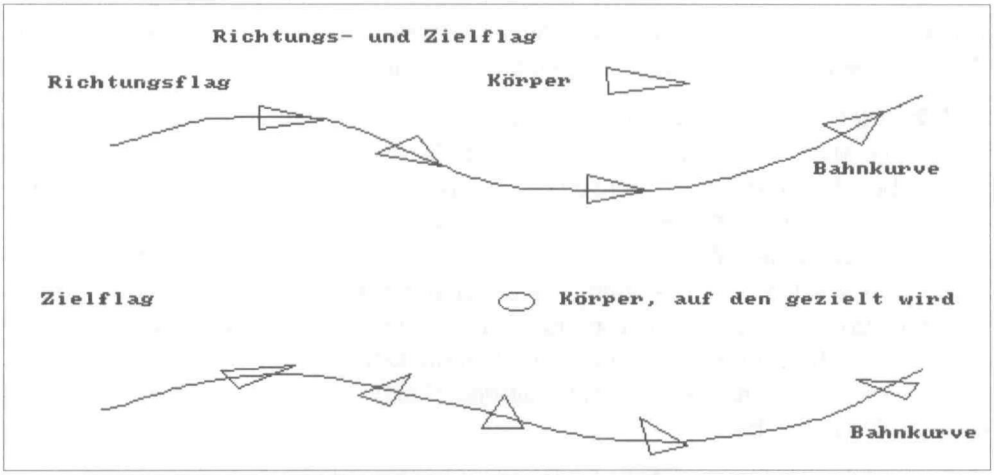


Bild 12: »Richtung« und »Ziel« beeinflussen die Ausrichtung von Körpern

Ziel: Mit dem Zielflag können Sie einen Körper auf einen anderen ausrichten. Er ist dann immer so orientiert, daß er auf den angegebenen Zielkörper zeigt. Besonders bei Kameraflügen ist das Zielflag eine unschätzbare Hilfe.

Die Wirkung dieser beiden »Flags« ist in Bild 12 angedeutet.

Beide »Flags« verändern die in den »Snaps« vorgegebene Richtung. Wenn das Programm in diesem Sinne frei über die Richtung der Körper entscheiden soll, müßte es normalerweise auch wissen, wo bei einem Körper vorne und hinten ist, damit es etwas Sinnvolles daraus machen kann. Da es das aber nicht wissen kann, muß der Anwender etwas nachhelfen: Der »Animator« hat eine standardmäßige »vorne«-Richtung, die er in Richtung der Bahnkurve dreht. Nur kann dann der Körper niemals rückwärts fliegen. Ein Wechseln der Flugrichtung ist so nicht möglich. Um auch diesen Fall einzubeziehen, nimmt der »Animator« die durch die »Snaps« vorgegebene Richtung und vergleicht sie mit der Bahnrichtung. Ist die Winkeldifferenz kleiner als 90 Grad, so wird der Körper vorwärts ausgerichtet (Bild 13).Wenn jedoch der Winkel 90 Grad übersteigt, dann nimmt der »Animator« an, daß der Körper rückwärts fliegen soll und dreht ihn entsprechend um (Bild 14).

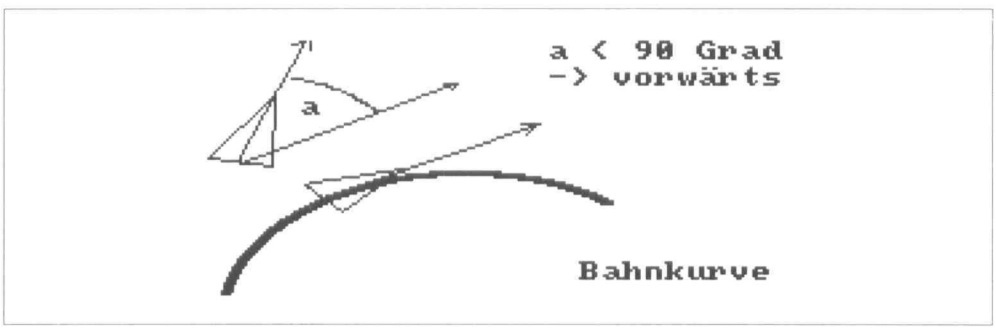


Bild 13: Das »Richtung«-Flag arretiert die Vorderseite des Körpers in Flugrichtung

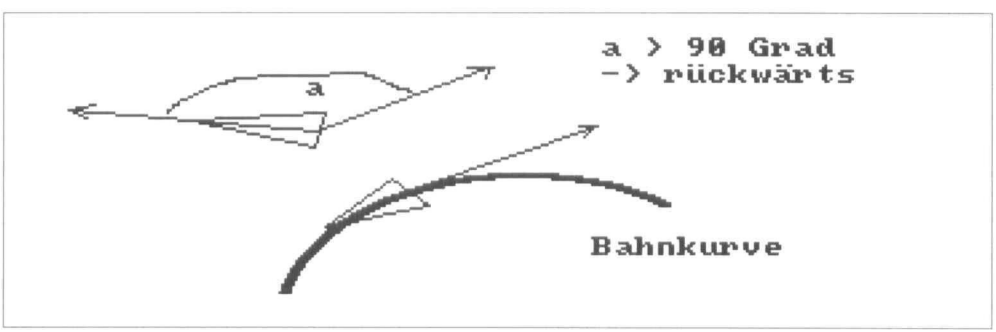


Bild 14: Sogar Rückwärts-Flüge sind auf diese Weise möglich

Sie sollten also, wenn Sie das Richtungs- oder Zielflag setzen, in den »Snaps« zumindest etwa die Richtung der Körper festlegen. So weiß der »Animator«, ob der Körper vorwärts oder rückwärts fliegen soll und interpoliert entsprechend (Bild 15).

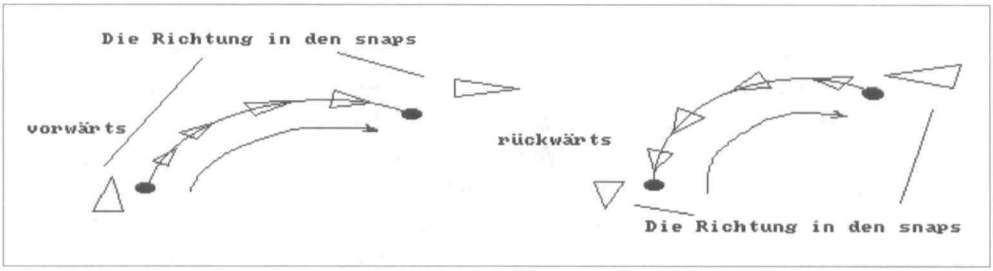


Bild 15: Die grobe Richtung muß noch immer der Regisseur angeben

Falls Sie das Hüpf-»Flag« eingeschaltet haben, dann hat die Richtungsinterpolation logischerweise keinen Sinn. In dem Fall gibt es keine Bahn und daher auch keine Bahnrichtung.

Take: Ein Take ist eine Sammlung von Sequenzen, die gleichzeitig abgearbeitet werden sollen, beispielsweise für eine Animation. Beim Bewegen und Interpolieren von Körpern wird ja stets nur mit der aktuellen Sequenz gearbeitet. Es ist aber auch interessant, zu sehen, wie mehrere Sequenzen oder Bewegungsabläufe zusammenwirken. Hier stellt der Anwender je nach Bedarf eine Menge von Sequenzen zusammen.

Take.Zeige: Mit diesem Untermenüpunkt wird im Auswahlfenster angezeigt, welche Sequenzen das Take gerade enthält. An der Farbe des Sequenznamens im Auswahlfenster können Sie die Art der Sequenz erkennen: Schwarz für eine Körper-Sequenz, Blau für eine Kamera-Sequenz und Rot für eine Licht-Sequenz. Klicken Sie dann mit rechts oder links ins Auswahlfenster, wenn Sie weitermachen wollen.

Take.define: Damit kann ein neuer Take definiert werden. Klicken Sie einfach so lange Sequenzen an, bis Sie alle beisammen habe, die gleichzeitig ablaufen sollen.

multi_seq: Dieses »Flag« hat auch mit dem Take zu tun. Normalerweise arbeitet der Anwender nur mit der aktuellen Sequenz, deren Name oben im Sequenzfenster angezeigt ist. Klicken Sie auf »intp«, wird nur der oder die Körper der aktuellen Sequenz interpoliert. Ist dagegen »multi_seq« gewählt (was durch einen Haken an diesem Menüpunkt zu erkennen ist), werden alle Sequenzen, die im Take enthalten sind, interpoliert. Das dauert dann je nach Anzahl und Umfang der Sequenzen länger, aber dafür haben Sie dann auch gleich die Kontrolle über das Zusammenwirken verschiedener Sequenzen. Wird »multi_seq« eingeschaltet, so zeigt der »Animator« das aktuelle Take an (sofern eines definiert ist) und fragt, ob es geändert werden soll. Wenn ja, verfahren Sie wie bei »Take.define«. Danach werden, wenn Sie interpolieren, neben der aktuellen Sequenz, auch noch alle Sequenzen des Takes mit interpoliert.

plot_pfad: Mit diesem »Flag« veranlassen Sie »Animator« den Bewegungspfad für den Körper der aktuellen als weiße Linie zu zeichnen. Gleichzeitig werden dann noch die

Positionen der »Snaps«, die im Sequenzfenster zu sehen sind, in der Kurve gekennzeichnet. Sie haben damit unmittelbare Sichtkontrolle, welcher »Snap« zu welcher Position im Raum gehört.

delete_sequenz: Dies ist zum Löschen einer Sequenz gedacht. Sie sollten grundsätzlich Sequenzen löschen, die Sie nicht mehr benutzen wollen – das spart Speicher und Rechenzeit. Außerdem sollten Sie für eine Animation (egal ob mit »Beams« oder »Scan«) die Sequenzen löschen, die nicht dazu gehören, und dann die restlichen Sequenzen in einer separaten Datei speichern.

3.13 Erzeugen von Testanimationen

Die Darstellung von Zwischenpositionen durch Interpolieren und die Wire-Frame-Animation lassen bereits ein ziemlich realistisches Urteil über das spätere Aussehen der Animation zu. Unstimmigkeiten in der Bewegung oder eine unpassende Bildkomposition lassen sich damit durchaus finden. Lediglich die Abstimmung der Farben oder die exakte Wirkung der einzelnen Körper in der Bewegung sind dabei unsichtbar. Um darüber Informationen zu gewinnen, gibt es keinen anderen Weg als eine Animation berechnen zu lassen. Da eine Animation bereits mit wenigen Bildern zu lang wäre, um sie mit »Beams« zu berechnen (mehrere Wochen Wartezeit, bevor man die nächste Korrektur anbringen kann, sind einfach zu viel), bietet es sich geradezu an, dafür das Programm »Scan« zu verwenden. Es ist schneller als »Beams«, dafür hat der mit »Scan« berechnete Film nicht die gleiche Qualität – ein Nachteil, den man bei einer Probeberechnung gern in Kauf nehmen kann.

Während eine Skelett-Animation noch in Sekundenbruchteilen errechnet wird (für die Berechnung geht meist nicht mehr Zeit drauf als für die Darstellung am Bildschirm) und auch die Wire-Frame-Animation höchstens die Zeit einer kurzen Kaffeepause in Anspruch nimmt, dauert die Testanimation mit »Scan« schon bedeutend länger. Sie dauert je nach Anzahl der Bilder zwischen einigen Stunden und wenigen Tagen.

Aus diesem Grund beschränkt sich das interaktive Korrigieren trotz aller Geschwindigkeitsvorteile von »Scan« gegenüber »Beams« sicherlich auf die beiden erstgenannten Methoden. »Scan« ist dagegen ausführlich im Kapitel 6 beschrieben.

Die Funktionen zum Erzeugen einer Wire-Frame- oder Skelett-Animation sind im Preview-Menü untergebracht. Die Vorgehensweise ist im Prinzip immer wieder die gleiche: Wenn eine Sequenz fertiggestellt ist (siehe Kapitel 3.12), wählt man einen der folgenden Menüpunkte, betrachtet das Ergebnis, korrigiert gegebenenfalls etc. Das Preview-Menü besteht aus drei Punkten: »make_anim«, »show_anim« und »delete_anim«.

Für jede Animation benötigt das Programm folgende Informationen:

- Welche Sequenzen sollen ausgeführt werden?

- Welches Zeitintervall soll animiert werden?
- In wie viele Einzelbilder soll das Intervall aufgeteilt werden?
- Aus welcher Blickrichtung betrachtet der Zuschauer (bzw. die Kamera) die Objekte?

Bei den Sequenzen nimmt der »Animator« von sich aus an, daß stets die aktuelle Sequenz animiert werden soll. So lange es noch keine Sequenz gibt, sind die drei Menüpunkte nicht wählbar. Sollen mehrere Sequenzen ausgeführt werden, so geht dies über »multi_seq« (siehe Kapitel 3.12).

Die Frage des Zeitintervalls ist bereits durch die aktuelle Sequenz gelöst. Es wird nämlich das Zeitintervall übernommen, das dort mit Hilfe des »intv«-Gadgets definiert wurde und als waagerechte rot-blau Linie sichtbar ist. Ist kein Intervall definiert, so wird einfach die ganze Sequenz vom ersten bis zum letzten »Snap« als Intervall angenommen.

Die Anzahl Einzelbilder erfragt das Programm vom Benutzer, sobald er einen der Menüpunkte gewählt hat. Für eine flüssige Animation sind mindestens 10 bis 20 Bildern pro Sekunde empfehlenswert – natürlich kann dies nur von Fall zu Fall entschieden werden, denn die Anzahl der Bilder hängt auch davon ab, wieviel sich in den einzelnen Bildern ändert. Der »Animator« bietet im Frage-Fenster eine Bildanzahl an, die etwa 10 Bildern pro Sekunde entspricht. Bei Bedarf kann diese Zahl aber jederzeit verändert werden.

Wird eine Kamera-Sequenz für die Animation verwendet, so kann dies ungeahnte Effekte hervorrufen. Beispielsweise ist es möglich, daß der »Animator« geduldig eine kompizierte und vollkommen richtig konstruierte Animation berechnet, das Ergebnis aber nur eine unveränderte Szene ist, die sich 50 Bilder lang nicht von der Stelle bewegt. Der Grund ist einfach: Die Animation wird immer aus der eingestellten Perspektive gezeigt. Ist aber eine der Paralleldarstellungen (»vorne«, »hinten« oder ähnliches) eingestellt, dann kann die Kamera fliegen, wohin sie will, der Betrachter wird die Szene weiterhin von vorne, oben oder rechts sehen. Daher sollten Sie bei einer Kamera-Sequenz vorher unbedingt überprüfen, ob die Perspektive auf die Kamera eingestellt ist (im »Plotten«-Menü der Punkt »persp« oder die Taste »p«). Denn nur dann wird die Animation so gezeigt, wie sie von der Kamera aus sichtbar ist.

Folgende Funktionen werden von den Menüpunkten übernommen:

make_anim: Diese Funktion stellt die Animation zusammen. Solange »multi_seq« ausgeschaltet ist, verwendet »make_anim« dafür nur die aktuelle Sequenz. Per Untermenü wird noch die Auswahl zwischen Skelett- und Wire-Frame-Animation angeboten.

make_anim.skelett: Hier fragt der »Animator« nach der Bildanzahl und legt dann sofort los. Im Statusfenster zählt er die Bildnummern durch. Nach Fertigstellung

erscheint ein neuer Bildschirm, auf dem die Animation abläuft. Klicken mit der linken Maustaste beendet die Animation und läßt den Screen wieder verschwinden. Die Skelettanimation ist nur zur schnellen Kontrolle eines Bewegungsablaufes gedacht und erfüllt daher keine besonderen Qualitätsansprüche, dafür wird sie sehr schnell erzeugt.

make_anim.Wire-Frame: Eine Wire-Frame-Animation verbraucht erstens einiges an Speicher (verglichen mit der Skelettanimation) und braucht auch deutlich länger zur Berechnung (mehrere Minuten). Deswegen werden die Animationsdaten hierbei auch in einer eigenen Datei abgelegt, so daß sie später jederzeit wieder verwendet werden können. Der »Animator« fragt deshalb an dieser Stelle nach einem Dateinamen. Danach wird wie bei der Skelett-Animation nach der Anzahl Bilder gefragt. Das Zeitintervall wird ebenfalls dem eingestellten Intervall der aktuellen Sequenz entnommen. Daraufhin will das Programm noch wissen, ob die Bilder während der Animationserzeugung gezeigt werden sollen. Wenn ja, kriegen Sie den Bildaufbau jedes einzelnen Bildes mit, was bei kurzen Animationen noch interessant ist. Ab einer Dauer von einigen Minuten ist es aber sicher sinnvoller, in der Zeit etwas anderes zu machen. Eine Wire-Frame-Animation wird von dem Programm »Play« gezeigt (siehe Kapitel 7). Dazu genügt es, vom Manager aus Play aufzurufen und den Namen der soeben erzeugten Datei anzugeben.

show_anim: Jede Skelettanimation kann so lange nochmals gezeigt werden, bis sie gelöscht oder mit einer neuen überschrieben ist. Dazu ist nicht mehr nötig als im Preview-Menü »show_anim« zu wählen.

Delete_anim: Diese Funktion betrifft nur die jeweils letzte erzeugte Skelett-Animation. Diese wird ohne Rückfrage gelöscht, d.h. der Speicher, der für die Animationsdaten reserviert war, wird wieder freigegeben. Wenn Sie die Funktion aus Versehen aufrufen, ist das auch kein Beinbruch, da die gleiche Skelett-Animation innerhalb kurzer Zeit erneuert werden kann. Außerdem ist bei Speichermangel zu empfehlen, allzu lange Skelett-Animationen unbedingt schnell wieder freizugeben. Wire-Frame-Animationen können hier nicht gelöscht werden. Sie sind als Dateien gespeichert und sollten daher auch per CLI-Kommando gelöscht werden, wenn sie nicht mehr gebraucht werden.

3|14| Skelette

Ein Skelett ist die vereinfachte Darstellung eines Körpers. Es sollte einfach genug sein, daß der Körper vom Programm möglichst schnell gezeichnet werden kann, aber auch detailliert genug, um dem Betrachter einen ungefähren Überblick über Lage, Orientierung und Ausdehnung des Körpers zu geben.

Bei »Construct« gibt es einen ähnlichen Ansatz bei der Definition der Kamera. Um die Kamera in Echtzeit zu bewegen, wird anstatt des darzustellenden Körpers ein minimal

umschließender Quader gezeichnet, eine »Bounding Box«. Das genügt, um sich über die Lage der Kamera relativ zum Körper zu orientieren.

Für Körper, die quaderförmig oder ähnlich sind, wären als Skelette Boxen genau das richtige. Kompliziertere Körper, Roboter, Flugzeuge oder ähnliches wären mit Boxen allerdings nur ungenügend dargestellt.

Der »Animator« hat aber weder Ahnung über die Formen von Körpern, noch über über die assoziativen Gewohnheiten des Anwenders, also überläßt er die Skelettbildung dem letzteren. Liest der »Animator« eine neue »Construct«-Datei mit unbekannten Körpern ein, so erzeugt er für diese je ein Standardskelett. Er kann zwischen drei Arten auswählen und erzeugt entweder eine Box, ein achsenparalleles Viereck oder eine achsenparallele Linie. Die Auswahl nimmt er anhand der Form der Bounding Box des neuen Körper vor. Ist sie lang und dünn, erzeugt er eine Linie, bei flachen Bounding-Boxen erzeugt er ein Rechteck und sonst einen Quader. In vielen Fällen reicht das aus, aber manchmal muß der Benutzer das vorgegebene Skelett nachbearbeiten.

Allerdings hilft die Körperbaumstruktur etwas, so daß einiges an Aufwand gespart wird. Lediglich Körper, die keine Söhne mehr haben, erhalten Skelette. Bei Körpern mit Söhnen besteht das Skelett logischerweise aus den zusammengesetzten Skeletten der Söhne.

Wenn beispielsweise ein Körper »Robot« vorliegt, der aus den Teilkörpern »linkes_bein«, »rechtes_bein«, »linker_arm«, »rechter_arm«, »rumpf« und »Kopf« besteht, so kann für Robot selbst kein Skelett bearbeitet werden. Für Beine und Arme reicht eventuell die vorgegebene Linie, nur für den Rumpf und den Kopf sollte man etwas Arbeit investieren.

Die dafür benötigten Werkzeuge befinden sich im Skelett-Menü. Durch Wählen des obersten Menüpunktes, »make_skelett«, öffnet man die anderen Menüpunkte, die vorher gesperrt waren. »Animator« fragt daraufhin, welcher Körper ein neues Skelett erhalten soll. Bequemerweise bietet er im Auswahlfenster nur die Körper an, die keine Söhne mehr haben. Nachdem der Anwender sich entschieden hat, wird der betreffende Körper als Plotkörper eingestellt und außerdem das Plotfenster auf die Ausmaße des Körper skaliert. Alle anderen Menüs (außer die für die Darstellungsarten) sind während dieser Zeit gesperrt.

Der erste Menüpunkt »alles_Linien« hat eine Sonderfunktion, die weiter unten erklärt wird.

Die folgenden Menüpunkte bearbeiten das Skelett des Körpers:

get_skelett: Mit dieser Funktion können Sie das Skelett eines beliebigen, anderen Körpers übernehmen. Dies bietet sich beispielsweise an, wenn mehrere gleiche Körper existieren. Für die braucht dann nur ein Skelett konstruiert werden. Für die anderen Körper wird es mit »get_skelett« übernommen.

box: Hiermit wird die Bounding-Box, der minimal umschließende Quader des Körpers, als Skelett erzeugt, und zusätzlich zum bisherigen Skelett dargestellt. Allerdings wird das Skelett gelöscht, das der Körper bisher hatte.

Fläche: Mit dieser Funktion wird für den Körper als Skelett ein flaches Viereck erzeugt.

Linie: Anstatt der ganzen Box, wird nur eine Skelettlinie über die längste Seite der Bounding-Box des Körpers erzeugt.

new_pt: Mit »new_pt« wird ein neuer Skelettpunkt erzeugt. Nach Anwahl der Funktion klickt man mit der Maus an die Stelle des Körpers, an die der neue Punkt gesetzt werden soll. Der Punkt wird in Breite und Höhe entsprechend der Maus gesetzt, und in der Tiefe in die Mitte des Körpers. Mit den »Vollbild«-Gagdet können Sie sich übrigens jederzeit den gewünschten Körper auf den Bildschirm holen.

new_line: Mit »new_line« wird eine neue Skelettlinie geschaffen, sozusagen ein Knochen des Skeletts. Daraufhin müssen Sie den Anfangs- und Endpunkt der Linie anklicken. Sie fahren jeweils mit der Maus in die Nähe der Skelettpunkte, die durch eine Linie verbunden werden sollen und klicken mit der linken Maustaste. »Animator« nimmt dann immer den Punkt, der sich am nächsten zum Mauszeiger befindet. Klicken Sie zweimal den gleichen Punkt an, wird der Vorgang abgebrochen.

move_pt: Mit »move_pt« können Sie einen Punkt verschieben. Fahren Sie mit dem Mauszeiger in die Nähe des Punktes, den Sie bewegen wollen, klicken Sie mit links, sodann folgt der Skelettpunkt und alle angeschlossenen Skelettlinien der Bewegung des Mauszeigers. Diese Funktion ist sehr hilfreich, um einen neuen Skelettpunkt, der ja anfangs nur in zwei Dimensionen richtig positioniert wurde, auch in der dritten an die richtige Stelle zu setzen. Haben Sie zuvor einen neuen Skelettpunkt bei Ansicht von vorne erzeugt, schalten Sie jetzt auf Seitenansicht um (mit den Plotfenstergadgets »links« oder »rechts«) und bewegen dann den neuen Punkt mit move_pt auch in die richtige Tiefe.

delete_pt: Dieser Menüpunkt dient zum Löschen eines Skelettpunktes. Bewegen Sie den Mauszeiger zu dem Punkt, der gelöscht werden soll und klicken Sie mit links. Der Punkt, sowie alle Skelettlinien, die an dem Punkt enden, werden daraufhin gelöscht.

delete_line: Das gleiche geht auch mit Skelettlinien. Sie müssen hierbei die beiden Endpunkte einer Skelettlinie anklicken, worauf die Linie gelöscht wird. Die angeklickten Endpunkte bleiben aber erhalten.

skelett_fertig: Wenn Sie mit der Bearbeitung fertig sind, wählen Sie diese Funktion. Daraufhin werden die anderen Menüs wieder freigegeben und das Skelettmenü gesperrt.

Sobald Sie für alle Körper, für die es nötig war, die Skelette bearbeitet haben, sollten Sie die Skelettinformationen in einer Skelettdatei speichern. Das geschieht im Projektmenü unter »save_skl«. Geben Sie dort einfach den Namen der »Construct«-Datei an, die der »Animator« geladen hat. »Animator« fügt automatisch ».skl« an den Namen an, beim nächsten Laden der Szene werden dann die neuen Skelette automatisch mitgelesen.

Noch ein Tip. Bei Skeletten ist weniger besser, als zuviel. Bedenken Sie, daß der Bildaufbau beim Bewegen von Körpern um so länger dauert, je umfangreicher das Körperskelett ist. Bei einfachen Körpern ist das nicht so wesentlich. Aber bei Körpern, die sich aus mehreren Hierarchie-Ebenen zusammensetzen, sollten sich die Söhne im Detaillierungsgrad (beim Skelett) lieber etwas bescheiden. Es ist beispielsweise wenig sinnvoll, bei einem Roboter jedem Teilkörper eine eigene Box zu geben.

Soll ein komplexer Körper mit vielen aufwendig konstruierten Einzelteilen vereinfacht werden, so gibt es einen sehr hilfreichen Menüpunkt:

alles_Linien: Bei Szenen, die sehr viele Körper haben, kann das Arbeiten mit dem »Animator« etwas mühselig werden, denn er muß oftmals alle Skelettpunkte berechnen. Und bei vielen Körpern sammelt sich schon einiges an Punkten an. Besonders, wenn jeder Körper noch ein umfassendes Skelett besitzt. Dann hilft nur noch eines: die Skelette müssen drastisch vereinfacht werden. Das einfachste Skelett besteht aus einer einzigen Linie. Das Programm verwandelt bei diesem Menüpunkt die Skelette aller Körper in Linien. Dann erkennt man zwar schlechter, wie die Proportionen in den Szenen verteilt sind, aber die Bewegungen gehen dann wieder ruckfreier. Es ist aber noch einfacher, wenn Sie sich in der Anzahl Körper beschränken und nur diejenigen definieren, die Sie auch bewegen wollen.

3.15 Gelenke

Gelenke sind ein zentrales Konzept des »Animator«, das viele Arbeitsvorgänge erleichtert. Mit Gelenken werden diese Vorgänge sozusagen an die menschliche Logik angepaßt. Denken Sie nur an einen Roboter, der aus mehreren Körperteilen besteht. Würde es keine Gelenke geben, dann wären diese Teile für den »Animator« stets völlig frei im Raum verteilt und es wäre Sache des Anwenders, dafür zu sorgen, daß sie in jedem Bild eine sinnvolle Position haben – eine heikle und fehlerträchtige Aufgabe. Dank des Gelenk-Konzepts ist es dagegen möglich, dem Programm einmal mitzuteilen, daß zwei Körper ein gemeinsames Gelenk haben sollen, und damit bleiben sie für den Rest der Animation verbunden.

In vielen vorangegangenen Kapiteln tauchten bereits Hinweise auf Gelenke auf. Die entsprechenden Funktionen wurden bislang ausgespart, mit dem Verweis, sie seien in diesem Kapitel ausführlich behandelt. Sie finden weiter unten eine Zusammenfassung dieser Funktionen, zunächst muß aber das Gelenkkonzept näher beleuchtet werden.

Gelenke sind Verbindungsglieder innerhalb von komplexen Körpern. Komplex heißt in diesem Zusammenhang, daß der Körper aus vielen Teilkörpern zusammengesetzt ist, die einen geometrischen Zusammenhang haben. Paradebeispiel hierfür ist ein Roboter. Er setzt sich aus vielen Teilkörpern zusammen, wie etwa Rumpf, Kopf, Beine, Arme etc. – je nach Detaillierungsgrad unterschiedlich viele. Diesen Roboter stets nur als Ganzes zu bewegen, wäre sehr unbefriedigend. Bewegt er sich von Punkt »A« nach Punkt »B«, so soll er nicht auf einer Schiene gleiten, sondern Schritt für Schritt den Weg zurücklegen.

Mit den bisher beschriebenen Mitteln könnte man auch für jeden Teilkörper eine eigene Sequenz bilden und die Bahnen aller Körper so zusammenstoppeln, daß insgesamt Schrittbewegungen des Roboters herauskommen: Viel Spaß mit dieser vermutlich mehrjährigen Kleinarbeit. Diese Arbeit wird wesentlich erleichtert, wenn zwischen den Körperteilen Gelenke sitzen, die dem Programm mitteilen, welche Körper zusammengehören, und an welcher Stelle sie miteinander verbunden sind. Mit dieser Information kann das Programm selbständig ermitteln, welche Körper mitbewegt werden müssen, wenn beispielsweise der Oberschenkel eines Roboters um 45 Grad gedreht wird. Der »Animator« achtet dann automatisch darauf, daß die beteiligten Körper in ihren Berührungspunkten zusammenbleiben und nicht etwa verrutschen.

Genau diese Informationen bekommt der »Animator« über Gelenke. Man kann sie sich genauso vorstellen wie menschliche Gelenke oder Scharniere einer Maschine. Genaugenommen sind sie Kugelgelenke, denn die angehängten Körper sind in jeder Richtung drehbar.

Leider kann man nicht beliebige Körper mit Gelenken verbinden. Das hängt wieder mit der Baumstruktur und den Koordinatensystemen zusammen. Nur Brüder (siehe Kapitel 3.4) dürfen untereinander über Gelenke verbunden werden. Brüder sind Körper, die einen gemeinsamen Vater haben. Das ist dann auch der Hauptgrund für diese Einschränkung. Man kann die Koordinaten von Brüdern über das Koordinatensystem des Vaters in Einklang bringen.

So einschränkend das Ganze auch klingt, so einfach kann man auch in »Construct« schon dafür sorgen, daß es nicht allzu belastend wird. Achten Sie einfach bei der Konstruktion darauf, welche Körper Sie später mit Gelenken verbinden wollen, und erzeugen Sie die Abhängigkeiten der Körper entsprechend.

Einige Beispiele verdeutlichen die Vorteile, die das Gelenk-Konzept bietet.

In Bild 16 sind zwei Körper »A« und »B« zu sehen, die über ein Gelenk »A-B« verbunden werden, das genau zwischen den beiden liegt. Es ist jetzt leicht möglich, einen der Körper um das Gelenk zu drehen. Man kann sich vorstellen, daß die Körper mit unsichtbaren Fäden oder Stäben mit dem Gelenk verbunden sind.

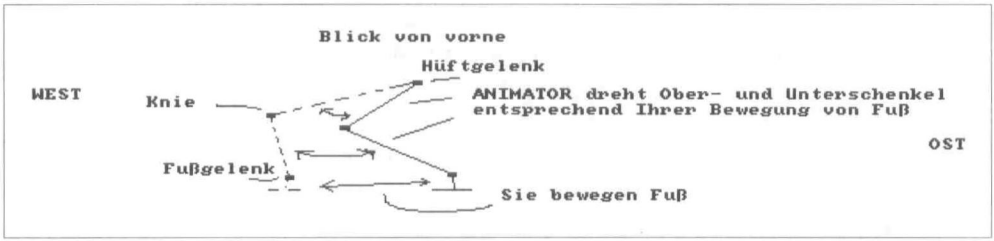


Bild 16: Ein einfaches Gelenk zwischen zwei Körpern

Bild 17 zeigt die selben Körper mit einem Gelenk, das außerhalb der beiden Körper liegt. Jedes Gelenk kann beliebig plaziert werden. Nur hat das beim Drehen die entsprechenden Auswirkungen – denn der Drehpunkt liegt immer am Gelenk.

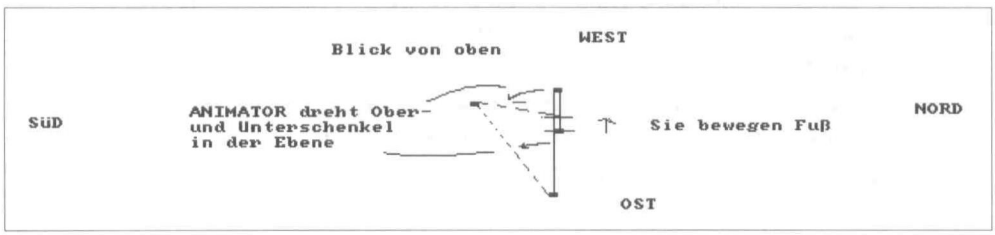


Bild 17: Gelenke können auch außerhalb eines Körpers liegen

Bild 18 zeigt die Verschiebung eines mit Gelenken verbundenen Körpers: Wenn »A« verschoben wird, muß »B« mit, da es an dem Gelenk hängt, das automatisch alle Bewegungen von »A« mitmacht.

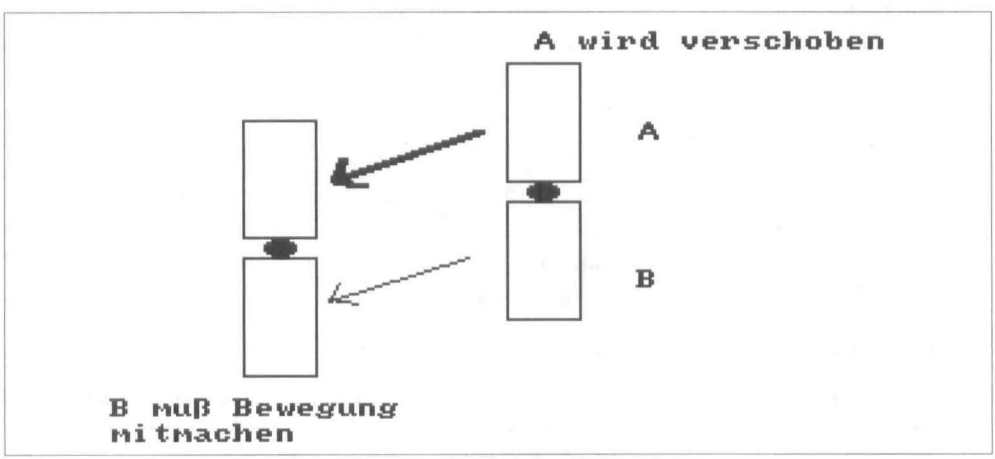


Bild 18: Die Verschiebung eines einfachen Gelenks

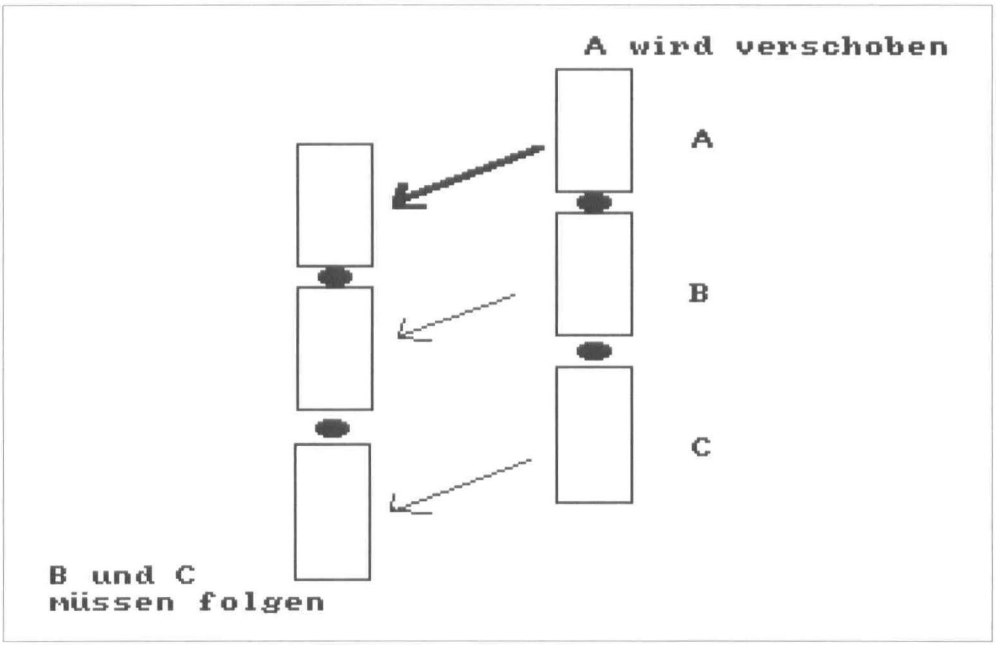


Bild 19: Auch komplexe Körper werden auf dieselbe Art bewegt.

Hängt an »B« dagegen noch ein Körper »C«, so ist auch der gezwungen, sich mitzubewegen, da er über ein Gelenk mit der Bewegung von »B« – und damit automatisch auch mit der von »A« – gekoppelt ist

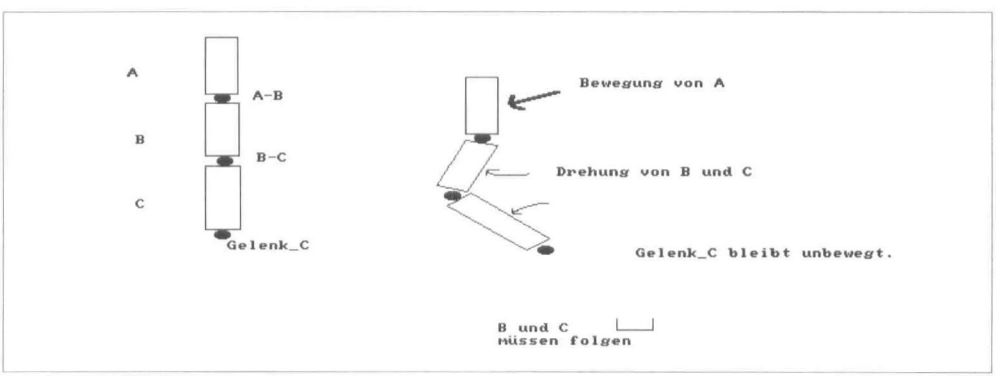


Bild 20: Kompensation durch Drehung.

Interessant wird es, wenn die Bewegung des Körpers »A« durch Drehung von drei anderen Körpern (»B«, »C« und »D«) kompensiert werden soll. Auch das ist mit dem »Animator« kein Problem. Bild 20 verdeutlicht diesen Sachverhalt.

Der Sinn dieser Variante wird erst dann deutlich, wenn man sich vorstellt, daß hinter »C« tatsächlich noch andere Körper hängen, beispielsweise der komplette Roboter. »A« könnte dann ein Fuß oder eine Hand sein. Durch die Bewegung von »A« werden nur die beiden nächstgelegenen Körper direkt beeinflußt. Das hat bei Gelenkkörpern große Vorteile. Sie können Hand oder Fuß bewegen, ohne daß gleich der Rest des Körpers mitgeschleift wird.

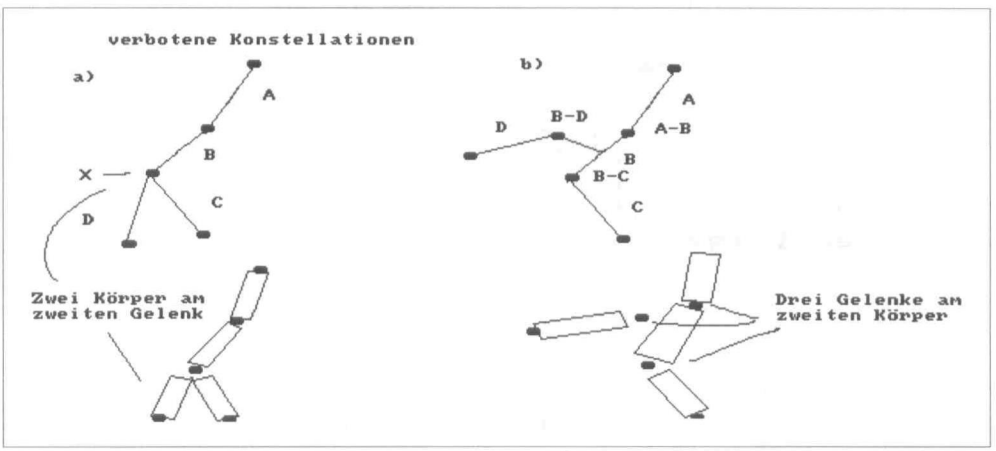


Bild 21: Einschränkungen des Gelenk-Konzepts

Wieso konnte die Drehung nicht schon im Beispiel Nr. 19 ausgeführt werden? Es gab dort noch keinen zweiten Drehpunkt für »C«. In Bild 20 ist es eindeutig, daß sich »C« nur um das Gelenk »Gelenk_C« drehen kann.

Allerdings gibt es dazu zwei Einschränkungen. Bild 21 stellt diese grafisch dar.

Die beiden Einschränkungen im Klartext:

- An »B« darf am Gelenk »X« nur ein weiterer Körper hängen. Also entweder »C« oder »D«, sonst ist eine Drehung von »B« keine erlaubte Bewegung.
- Ebenso darf »B« außer den Gelenken »A-B« und »B-C« keine weiteren besitzen, an dem andere Körper hängen. Auch hier wäre die Drehung nicht eindeutig definierbar.

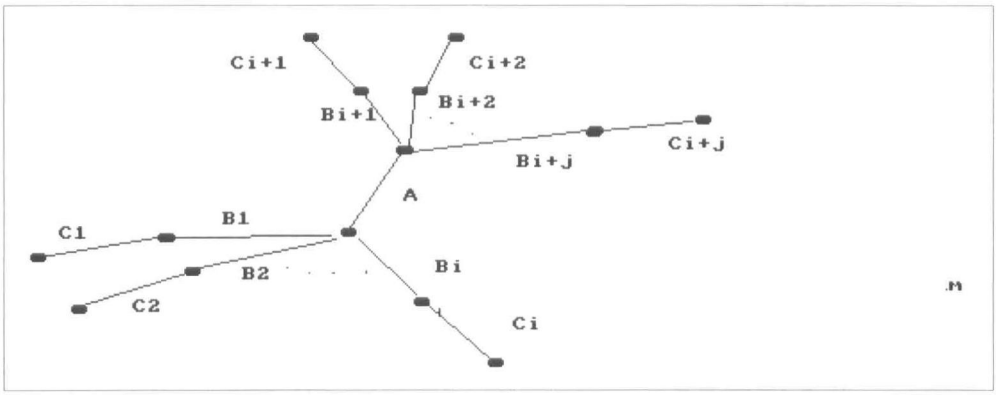


Bild 22: Bedingungen für den Ausgleich durch Drehung

Bild 22 zeigt die Bedingungen, die erfüllt sein müssen, damit der »Animator« die Bewegung eines Körpers »A« durch Drehung seiner Nachbarkörper ausgleichen kann. Im folgenden definieren wir einen Nachbarkörper von »A« als einen, der mit »A« über ein Gelenk verbunden ist.

Die beiden Bedingungen:

- Für alle Nachbarkörper »B_i« von »A« muß gelten: »B_i« hat genau ein weiteres Gelenk, an dem genau ein weiterer Körper »C_i« hängt.
- Für die Nachbarkörper »C_i« von »B_i« gilt: »C_i« hat genau ein weiteres Gelenk. Wie viele Körper daran hängen, ist gleichgültig.

Wie viele Nachbarkörper »A« hat, ist im Prinzip egal. Sie können sich aber sicher denken, daß mit steigender Anzahl der Rechenaufwand stark ansteigt. Das äußert sich dann durch starkes Ruckeln beim Bewegen von Körpern im Plotfenster.

Wenn die Bedingungen für einen oder mehrere Nachbarkörper nicht erfüllt werden, bewegt der »Animator« die betreffenden Körper einfach starr mit wie in Bild 19 dargestellt.

Eine Körperbewegung kann natürlich nicht beliebig kompensiert werden. Wenn Sie in Bild 20 den Körper »A« anstatt nach links unten nach oben ziehen, kann das durch keine Drehung von »B« und »C« ausgeglichen werden. Vielmehr werden dann die angehängten Körper einfach mitgezogen, als wäre die Drehung nicht möglich.

3.15.1 Gelenke im Sequenz-Menü

Viele Manipulationen von Gelenken geschehen über den »Gelenke«-Menüpunkt im Sequenz-Menü. Dieser Menüpunkt ist so lange gesperrt, bis Gelenke vorhanden sind. Wenn eine neue Sequenz (beispielsweise mit »neu_seq.Körper«) eröffnet wird, prüft der »Animator«, ob der Körper Brüder hat. (Das gleiche geschieht, wenn per Maus ein neuer »Work«-Körper ausgesucht und auf die Frage des Programms nach einer neuen Sequenz mit ja geantwortet wird.)

Wenn diese Bedingung erfüllt ist, dann fragt das Programm, mit welchen der vorhandenen Brüder er verbunden werden soll. Im Auswahlfenster können diese angeklickt werden. Wurden Gelenke ausgewählt, so wird jetzt das Gelenke-Untermenü freigemacht und die anderen Punkte gesperrt. Jetzt ist der Zeitpunkt, die Gelenke zu definieren.

Praktischerweise erzeugt der »Animator« an dieser Stelle eine neue Plotliste, die nur die soeben gewählten Körper enthält. Mit den Untermenüpunkten bestehen dann folgende Möglichkeiten:

neu_gelenk: Definiert ein neues Gelenk. Zunächst möchte der »Animator« wissen, welche Körper mit dem Gelenk verbunden werden sollen. Dies wird wie üblich über das Auswahlfenster eingegeben. Der Anwender kann im Prinzip beliebig viele Körpernamen angeben. Ein Gelenk kann auch mehr als zwei Körper verbinden. Empfehlenswert ist jedoch, sich hier auf zwei Gelenke zu beschränken. Erstens werden Bewegungen bei zu vielen Körpern an einem Gelenk völlig unübersichtlich und zweitens verlangsamt es die Handhabung, da das Programm dann um so mehr zu rechnen hat. Ein Gelenk kann auch nur einen Körper statt zwei oder mehr haben. Das ist beispielsweise sinnvoll, wenn man das Gelenk nur als Drehpunkt nutzen möchte. Als nächstes will das Programm den Namen des Gelenks wissen. Je nach Situation kann es sinnvoll sein, ein Gelenk nach seiner Funktion – beispielsweise »Knie«, »Scharnier« oder ähnliches – zu benennen oder nach den Namen der mit dem Gelenk verbundenen Körper. Die letztere Variante wird auch von »Animator« vorgeschlagen. Als letztes wird das Gelenk auf dem Plotfenster positioniert. Bewegen Sie dazu den Mauszeiger an die Stelle, wo das Gelenk sein soll und klicken Sie mit der linken Maustaste. Es empfiehlt sich, als Position die Verbindungsstelle der Körper zu verwenden, die über das Gelenk verbunden werden. Aber Sie können die Gelenke natürlich willkürlich in die Gegend setzen, dann wird die spätere Handhabung allerdings etwas verwirrend sein. Sie sollten sich diese Möglichkeit für später aufheben, wenn Sie Erfahrung mit Gelenken haben.

HINWEIS: Wenn schon mehrere Gelenke definiert sind, kann es vorkommen, daß der »Animator« ein weiteres ablehnt. In dem Fall ist im Netz der verbundenen Körper ein Zyklus entstanden, was nicht gestattet ist. Beispiel: Es wurden Gelenke zwischen Fuß

und Unterschenkel, Unterschenkel und Oberschenkel gesetzt. Jetzt soll – möglicherweise versehentlich – ein Gelenk zwischen Oberschenkel und Fuß gesetzt werden. Dies würde eine Rekursion erzeugen, was vom Programm abgelehnt wird.

Bewege_Gelenk: Genau wie bei den Skelettpunkten kann man die Gelenke nur in zwei Dimensionen positionieren. Um also auch die korrekte Position des Gelenks in der Tiefe zu erhalten, muß die Szene zunächst um 90 Grad gedreht werden. Stimmt die Position des Gelenks in dieser Dimension noch nicht, kann sie mit »Bewege_gelenk« korrigiert werden. Die Auswahl des zu bewegenden Gelenks erfolgt hier ebenso wie beim »Work«-Körper: Sobald der Mauszeiger in die Nähe eines Gelenks kommt, wird dessen Name in der Status-Zeile angezeigt. Wenn es der richtige ist, genügt ein Druck auf die linke Maustaste und das Bewegen der Maus an die neue Position – dort wo die Maustaste losgelassen wird, setzt der »Animator« die neue Position des Gelenks.

Delete_Gelenk: Sollten Sie sich mit einem Gelenk einmal gründlich geirrt haben – beispielsweise bei der Angabe der damit verbundenen Körper – so ist das möglicherweise nur noch mit Löschen des Gelenks zu lösen. Eben mit »Delete_Gelenk«.

Take_Gelenke: Bei umfangreichen Körpern kann das Definieren der Gelenke etwas länger dauern; nicht gerade Stunden, aber beispielsweise bei einem Roboter sind Sie schon einige Minuten beschäftigt. Wollen Sie nun eine neue Sequenz mit dem gleichen Roboter eröffnen, wäre es lästig, die gleichen Gelenke nochmals anzugeben. Mit »Take_Gelenke« können die Gelenke von anderen Sequenzen übernommen werden. Es werden dabei natürlich nur Gelenke angenommen, deren Körper auch zur neuen Sequenz passen. Alle zuvor hier definierten Gelenke werden dann allerdings gelöscht. Wenn noch weitere Gelenke definiert werden sollen, muß das danach geschehen.

Abbruch/Gelenke_fertig: Diese Funktion bricht die Gelenkdefinition ab oder schließt sie ab. Anfangs ist nur das Abbrechen gestattet. In dem Fall werden alle bis dahin definierten Gelenke und die neu eröffnete Sequenz gelöscht. Das bietet sich an, sollte man sich heftig geirrt haben oder aus sonstigen Gründen neu anfangen wollen. Abschließen kann das Programm die Gelenkdefinition erst, wenn alle Körper, die anfangs angegeben wurden, miteinander verbunden sind. Sobald das zutrifft, ändert sich der Text des Menüpunktes von »Abbruch« zu »Gelenke_fertig«. Das heißt nicht, daß jeder Körper mit jedem anderen nun eine gemeinsames Gelenk haben muß. Vielmehr muß man von jedem Körper über Gelenke jeden anderen erreichen können, eventuell über mehrere Schritte. Beispiel: Sie haben zu Beginn »neu_seq« mit dem Körper »A« gewählt. Auf die Frage nach den zu verbindenden Körpern haben Sie »B«, »C«, »D« und »E« angegeben. Sie haben Gelenke zwischen »A« und »B«, »B« und »C« und zwischen »D« und »E« gesetzt. Der »Animator« gibt jetzt den Menüpunkt »Gelenke_fertig« noch nicht frei, denn die Körpergruppen »A«, »B«, »C«, »D« und »E« haben noch keine Verbindung miteinander. »Animator« wartet so lange, bis Sie auch diese Lücke schließen. Eine Verbindung zwischen »A« und »C« existiert indirekt über

»B«. Hier muß also noch ein Gelenk beispielsweise zwischen »A« und »D« oder »B« und »E« erzeugt werden, dann ist der »Animator« zufrieden. Mit »Gelenke_fertig« werden auch die anderen Menüs wieder freigegeben.

Eine Sequenz mit Gelenken kann somit, im Gegensatz zu anderen Sequenzen auch mehrere Körper gleichzeitig enthalten. Der erste Körper, der bei der Sequenz angegeben wird (bei »neu_seq«), ist der Körper, der die Bahn aller anderen Körper der Sequenz bestimmt. Das ist der »Hauptkörper« der Sequenz. Da alle Körper miteinander verbunden sind, können sie nicht alle eine eigene Bahnkurve haben, sondern folgen, durch die Gelenke mitgezogen der Bahn des Hauptkörpers. Bei jedem »Snap« wird in diesem Fall die Position nur für den Hauptkörper gespeichert. Für alle anderen Körper wird nur noch die Orientierung vermerkt, d.h. die Winkel gegenüber der Grundorientierung. Der Hauptkörper bestimmt also die Bahnkurve der Körper. Sie sollten sich daher genau überlegen, welchen der Körper Sie als Hauptkörper wählen. Es empfiehlt sich meistens, einen relativ zentralen Körper als Hauptkörper zu bestimmen. Beim Roboter wäre der Körper »Rumpf« angebracht. Der bewegt sich relativ ruhig, während die Extremitäten, beispielsweise die Füße eher wild herumwedeln. Würde man einen Fuß als Hauptkörper benutzen, hätte das eine ziemlich unruhige Bahn des Roboters zur Folge.

3.15.2 Gelenke in Gadgets

Drei Gadgets im Plotfenster werden erst aktiv, wenn eine Sequenz mit Gelenken aktiv ist: »bewege_starr«, »drehe_Gelenk« und »drehe_starr«.

bewege_starr: Hier werden alle Gelenke der Sequenz eingefroren. Dadurch folgen alle Körper der Sequenz jeder Bewegung starr. Diese Funktion ist dann hilfreich, wenn man alle Körper der Sequenz bewegen möchte, deren Orientierung aber erhalten bleiben soll. Die Bewegung erfolgt ansonsten genau wie beim Gadget »bewege_normal«.

drehe_gelenk: Hier wird der »Work«-Körper um eines seiner Gelenke gedreht. Hängt der Körper nur an einem Gelenk, dann geht es sofort weiter. Hängt er dagegen an mehreren, dann fragt das Programm vorher noch, um welches seiner Gelenke der »Work«-Körper gedreht werden soll. Man sollte dann einen Körper um ein Gelenk drehen, wenn man vermeiden will, daß Körper, die auf der anderen Seite des Gelenks liegen, durch die Drehung beeinflußt werden. Wenn bei einem Roboter ein Oberschenkel gedreht wird, die Hüfte aber unbewegt bleiben soll, so wählt man »drehe_gelenk« für den Oberschenkel und gibt als Gelenk das Hüftgelenk an. So werden dann zwar Unterschenkel und Fuß mitgedreht, Hüfte, Rumpf etc. bleiben aber unbewegt.

drehe_starr: Hier liegt der gleiche Fall wie bei »bewege_starr vor«, nur wird gedreht statt bewegt. Alle Körper werden um einen Drehpunkt starr gedreht, d.h. in der Lage

der Körper zueinander ändert sich nichts (es sind selbstverständlich jeweils nur die Körper der Sequenz gemeint). Der »Animator« nimmt die Position des Mauszeigers beim ersten Klicken als Drehpunkt, wartet, bis Sie ein zweitesmal klicken, dreht dann die Körper der Sequenz starr entsprechend der Mausbewegung, bis Sie ein drittesmal mit rechts klicken.

Leider bringen Gelenke auch Einschränkungen. Solange eine Gelenk-Sequenz aktiv ist, sind beispielsweise die Gadgets zum Vergrößern und Verzerren gesperrt. Diese beiden Punkte sind in der vorliegenden Version noch nicht implementiert. In der Entwicklung traten bislang zu große Schwierigkeiten bei der Realisierung von Vergrößerungen und Verzerrungen an den Gelenken auf.

3.15.3 Die Eigenschaften im Körper-Menü

Ein Punkt aus dem Körper-Menü wurde bei dessen Beschreibung ausgelassen: »Eigenschaften«. Dieser wird erst im Zusammenhang mit Gelenken wirklich interessant. Daher im folgenden die Beschreibung.

eigenschaften: Diese Funktion dient ebenfalls zur Manipulation von Gelenkkörpern. Der Anwender legt hier für ausgewählte Körper der Sequenz fest, wie eine Veränderung benachbarter Körper auf sie wirken soll. Ein gutes Beispiel ist Bild 29: Für den Roboter soll ein Schritt definiert werden. Beide Füße sind auf dem Boden. Das »bewege.normal«-Gadget bewegt den rechten Fuß an seine neue Position. Die angehängten Körper folgen der Bewegung durch Drehen. Der linke Fuß bleibt an seiner Position, was für den Schritt zunächst erwünscht ist (b). Bei einem zu weiten Schritt allerdings kann die Bewegung nicht mehr ausgeglichen werden und der linke Fuß rutscht mit (a). Um dem entgegen zu wirken, kann man für einzelne Körper Eigenschaften festlegen. Im Beispiel wäre es angebraucht, dem linken Fuß die Eigenschaft »bewege« kurzzeitig zu entziehen. Ist der Schritt dann zu weit, wird die Bewegung nicht ausgeführt. Jedem Körper kann kurzzeitig verboten werden, sich zu bewegen. Dies macht nur Sinn bei Gelenken, bei der die Bewegung eines Körpers auch andere mitziehen kann. Neben dem Bewegen kann man einem Körper auch das Drehen verbieten, was nützlich ist, wenn ein Körper zwar mitgezogen werden darf, seine Orientierung aber keinesfalls verändern soll. Die Eigenschaften können auf zweierlei Art manipuliert werden:

- Rechts neben den Gadgets zum Verändern sind noch zwei weitere mit Namen »beweg« und »winkel«. Über diese Gadgets können die jeweiligen Eigenschaften für den »Work«-Körper ein- bzw. ausgeschaltet werden.

- Beim Wählen von »Eigenschaften« im Körper-Menü erscheint das Auswahlfenster, in dem alle Körper der Sequenz aufgelistet sind. Hinter jedem Körper sind noch zusätzlich zwei Buchstaben dargestellt, die die augenblicklichen Eigenschaften des

Körpers symbolisieren. Die zwei Buchstaben stehen für Bewegen (»B«) und Drehen (»D«). Sie können für einen Körper festlegen, ob er sich drehen und/oder bewegen darf.

Die Liste könnte beispielsweise folgendermaßen aussehen:

```
RumpfB »D« Tür1B . Tür2B . Teil-a . .
```

Und wäre folgendermaßen zu interpretieren: Der Rumpf darf drehen und bewegen. Tür1 und Tür2 dürfen sich nur bewegen. Teil-a darf keines von beiden.

Als Voreinstellung haben alle Körper einer Sequenz beide Eigenschaften. Eigenschaften lassen sich einfach per Mausklick auf den entsprechenden Buchstaben umschalten. Ist eine Eigenschaft ausgeschaltet, so erscheint anstelle des Buchstabens ein ».«. Mit der zweiten Variante können gleichzeitig auch die Eigenschaften anderer Körper beeinflußt werden und nicht nur den »Work«-Körper. Der Menüpunkt »clear_eigenschaften« schaltet für alle Körper der Sequenz auf die Voreinstellung zurück, d. h. beide Eigenschaften sind wieder erlaubt. Das gleiche passiert, wenn die Sequenz gewechselt wird.

3.15.4 Take_Snaps

Die Funktion »take_Snaps« im Sequenz-Menü wurde schon im Kapitel 3.11 erläutert. Im Zusammenhang mit Gelenksequenzen muß dabei allerdings einiges zusätzlich beachtet werden. Bei einer Sequenz, die mehrere Körper enthält, welche über Gelenke miteinander verbunden sind, besteht jeder »Snap« aus zwei Sorten von Informationen:

- Für alle an der Sequenz beteiligten Körper die Orientierung im Raum.
- Für den Hauptkörper zusätzlich die Koordinaten.

Sollen die »Snaps« einer anderen Gelenksequenz übernommen werden, so macht das nur Sinn, wenn man allen Körpern der ersten Sequenz eindeutig Körper der zweiten Sequenz zuordnen kann. Der »Animator« muß beim Übernehmen eines »Snaps« einer fremden Sequenz wissen, welchen Körpern der aktuellen Sequenz er die Daten des »Snaps« zuordnen soll. Diese Zuordnung kann er aber nur bei Körpern vornehmen, die entweder genau gleich oder Doppelgänger im Sinne der Definition aus Kapitel 3.4 sind. Dies klingt kompliziert, in der Praxis kann man diese Bedingungen aber leicht einhalten.

Das Übernehmen von »Snaps« einer fremden Gelenksequenz ist nur sinnvoll, wenn beide Sequenzen hinsichtlich ihrer Körper und Gelenke übereinstimmen. Es macht wenig Sinn, die »Snaps« einer Robotersequenz für eine Hubschraubersequenz zu übernehmen. Das Übernehmen der »Snaps« einer bestimmten Sequenz ist dagegen bei Doppelgängern sehr nützlich. Eine denkbare Anwendung sind zwei Roboter, die

nebeneinander gehen o.ä. Wie man Doppelgänger erzeugt, ist in Kapitel 3.4 beschrieben. Die Körpernamen der Doppelgänger lauten genau wie die der Originale, mit zusätzlich einem /1 hintendran (bzw. /2, /3..., wenn Körper mehrfach verdoppelt sind).

3.15.5 Gelenkfunktionen im Special-Menü

Das »Special-Menü« enthält drei Menüpunkte für den Umgang mit Gelenken: »int_quick«, »int_good« und »show_gelenk«. Während die ersten beiden eine Einstellung vornehmen, ist der dritte Punkt für das Finden von Gelenken nach ihrem Namen zuständig.

show_gelenk: Dieser Menüpunkt zeigt eine Liste der definierten Gelenke und zeigt das Angeklickte markiert an.

Die beiden anderen Menüpunkte – int_quick und int_good – haben eine etwas andere Funktion: Sie treffen Voreinstellungen für die Interpolation von Gelenkbewegungen.

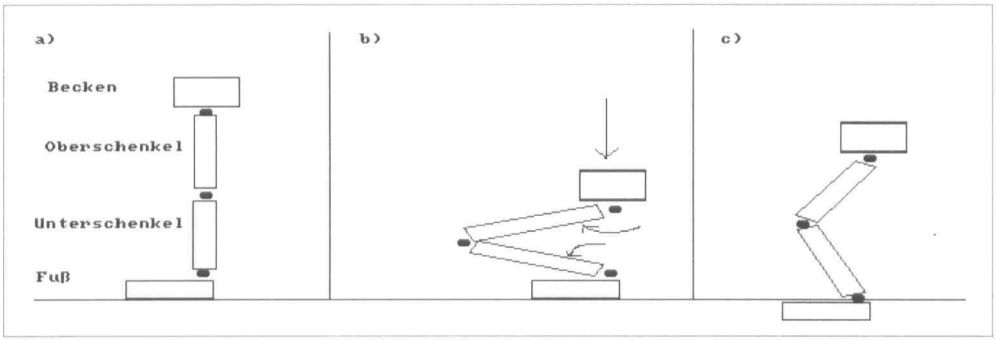


Bild 23: Die Schrittbewegung eines Roboters

Dabei stellt »intquick« eine Variante dar, die etwas schneller rechnet, dafür aber teilweise nachlässiger mit den Positionen der Körperteile umgeht (siehe unten). »int_good« dagegen berechnet die Position sehr genau, nimmt sich dafür allerdings auch etwas mehr Zeit.

Ein Beispiel verdeutlicht das eben Beschriebene sehr schnell: Ein Roboter, der auf einem Boden läuft, sollte mit den Füßen möglichst Bodenkontakt haben. Bild 23 stellt eine solche Figur dar. Dabei sind nur Becken, Oberschenkel, Unterschenkel und Fuß sichtbar.

Der Ablauf geht folgendermaßen vor sich:

- Ein Roboter steht aufrecht mit beiden Beinen fest auf dem Boden.
- Der Roboter ist in die Knie gegangen – beispielsweise weil das Becken vorsichtig herunterbewegt wurde. Der Fuß hat dabei seine Position nicht verändert.
- Das Interpolieren der Zwischenposition ergibt allerdings, daß der Fuß kurzfristig in den Boden eingesunken ist.

Die Erklärung dafür ist einfach: Der »Animator« hat wunschgemäß die Position des Hauptkörpers (in diesem Fall das Becken) interpoliert. Für die anderen Körper werden keine eigenen Position interpoliert, sondern nur ihre Orientierung, also ihre Winkel gegenüber der Grundorientierung. Und da zeigt sich, daß bei Interpolation der Winkel die daraus resultierende Position des Fußes nicht exakt mit der Anfangs- und Endposition übereinstimmt. Würde eine Animation dieses Beispiels berechnet, so würde der Fuß des Roboters in den Boden einsinken und sobald er die Endposition erreicht hat, wieder herauskommen. Daraus entsteht dann ein leicht federnder Gang.

Man kann nun entweder mühsam immer wieder Zwischenschritte interpolieren, dort die Position des Fußes korrigieren und dann für diese Position einen zusätzlichen »Snap« anlegen. Bei längeren Animationen kann dieses Vorgehen aber ziemlich langwierig werden. Die bessere Wahl ist sicherlich »int_good«. Ist diese Berechnungsvariante eingestellt, so werden automatisch korrekte Zwischenpositionen interpoliert. Dieser Vorgang braucht allerdings naturgemäß mehr Zeit und man sollte ihn sparsam einsetzen, beispielsweise bei Wire-Frame-Animationen oder bei Feinarbeiten, um Schritte genau zu positionieren.

3.16 Lichtsequenzen

Bisher war größtenteils von Körper- oder Kamera-Sequenzen die Rede. Daß auch Lichtquellen in Sequenzen ihre Eigenschaften ändern können, wurde nur einmal in Kapitel 3.11 kurz angesprochen (bei der Erklärung der Funktion »neu_seq«). Dieses Kapitel behandelt die Licht-Sequenzen ausführlich.

Jede Lichtquelle ist durch ihre Position und ihre Farbe gekennzeichnet. Daß die Position sich, wie bei Körpern oder Kameras in einer Sequenz manipulieren läßt, ist nicht weiter erklärungsbedürftig. Im Unterschied zu einem Körper, benötigt eine Lichtquelle keinerlei Informationen über ihre Orientierung und Größe in der Sequenz. Diese Informationen sind für Lichtquellen ohne Bedeutung. Sie strahlen in alle Richtungen gleich aus, folglich ist die Orientierung eine unnötige Information. Jede Lichtquelle ist per Definition ein Punkt. Und ein Punkt bleibt ein Punkt, also kann es auch keine Größe einer Lichtquelle geben.

Statt dessen ist die Lichtfarbe eine wertvolle Information. Es kann ein reizvoller Effekt sein, die Farbe einer Lichtquelle im Verlauf einer Animation sanft zu ändern. Genau das wird von Licht-Sequenzen unterstützt. Zu jedem »Snap« wird automatisch auch die aktuelle Farbe mitgespeichert – natürlich nur wenn sie vorher geändert worden ist.

Nach dem Wählen des Untermenüs »licht« vom Menüpunkt »Sequenz.neu_seq« fragt das Programm, welche der maximal fünf möglichen Lichtquellen zu der Sequenz gehören soll. Die Nummer der Lichtquelle ist gleichgültig, nur sollten bei mehreren parallelen Lichtsequenzen verschiedene Nummern vergeben sein. Anschließend will das Programm noch wissen, zu welchem Körper die Lichtquelle gehören soll. Ist dabei »all« oder kein Körper angegeben, dann bewegt sich die Lichtquelle im Weltkoordinatensystem. Wird hingegen ein anderer Körper ausgewählt, so beziehen sich die Koordinaten der Lichtquelle auf das Koordinatensystem des Körpers. Der Körper ist sozusagen der Vater der Lichtquelle geworden.

Der Sinn dieses Vorgehens ist das automatische Mitbewegen der Lichtquelle mit dem Körper. Dies ist besonders praktisch bei Körpern, die Lampen darstellen oder enthalten (Tischlampen, Autos mit Scheinwerfern etc.). Man braucht sich nicht um die korrekte Bahn der Lichtquelle bemühen, wenn sich der entsprechende Lampenkörper bewegt. Die Lichtquelle muß nur einmal in die richtige Position im Körper bewegt werden (anschließend »Snap« klicken), danach folgt die Lichtquelle automatisch allen Veränderungen des Körpers.

Die Lichtquelle wird im Plotfenster als weiße Raute dargestellt. Sie ist anfangs im Mittelpunkt der Szene positioniert und hat die Farbe weiß (1.0,1.0,1.0). Wie bei anderen Sequenzen können auch Lichtquellen bewegt und mit Flags versehen werden. Außerdem ist bei Lichtsequenzen im Körper-Menü der Untermenüpunkt Lichtfarbe wählbar.

Lichtfarbe: Beim Wählen dieses Menüpunkts öffnet sich ein Fenster mit vier Reglern, je einer für Rot, Grün und Blau, sowie für die Helligkeit. Im Gegensatz zu »Construct« kann der Anwender hier also die Farbe frei einstellen und muß sich nicht nach einer Materialfarbe richten. Die Farbe 1 im »Animator«-Screen (damit werden beispielsweise die Fensterumrandungen gezeichnet) ändert sich dann entsprechend der Farbeinstellung.

3.17 Das Projekt-Menü

In diesem ganz links im Menübalken angeordneten Menü sind alle Funktionen zum Laden und Speichern von Dateien zusammengefaßt. Sobald der »Animator« eine Datei lesen oder schreiben will, öffnet er einen File-Requester, der weitgehend dem aus »Construct« bekannten entspricht. Zusätzlich warnt das Programm, wenn auf eine bereits existierende Datei geschrieben werden soll.

Folgende Funktionen stehen zur Verfügung:

Load_skl: Hiermit können zusätzliche Skelett-Dateien geladen werden. Normalerweise enthält eine Szene nur die Skelett-Datei, die zur Grunddatei gehört. Beispielsweise könnte es aber sein, daß man für einem Körper »Tisch« ein Skelett erzeugt und gespeichert hat (in einer Datei namens »Tisch.skl«). Jetzt soll der Tisch mit anderen Körpern kombiniert werden. Die Skelett-Daten wären jetzt normalerweise für die neue Datei verloren. Anstatt für den Tisch das gleiche Skelett nochmals zu erzeugen, kann die alte Skelett-Datei mit »Load_skl« dazugeladen werden. Es können aber nur Skelette von Körpern übernommen werden, die geometrisch nicht verändert wurden.

Save_skl: Zum Speichern von Skelett-Dateien, die in einer anderen Szene weiterverwendet werden sollen, ist dieser Menüpunkt nötig.

Load_seq: Dieser Menüpunkt dient zum Lesen von Sequenz-Dateien. Hiermit können Sie Sequenzen lesen, die Sie zuvor in einer Sequenz-Datei abgespeichert haben. Es können aber nur Sequenzen von Körpern gelesen werden, die auch übereinstimmen. D.h. die Körper dürfen geometrisch nicht verändert sein.

Save_seq: Der umgekehrte Fall, das Speichern von Sequenzen. Es empfielt sich, Sequenzen regelmäßig zu speichern, da sie für die spätere Weiterbearbeitung durchaus eine hilfreiche Grundlage bieten.

Save_sz: Es ist zwar nicht möglich, »Construct«-Szenen nachzuladen, aber momentane Konstellationen können als »Construct«-Datei gespeichert werden, beispielsweise, um sie für einen Test oder einen Vorabeindruck als Bild zu berechnen.

Load_kam/li: Zum Laden von Kamera- und Lichtdateien.

Save_kam/li: Zum Speichern derselben.

fertig: Dieser Menüpunkt verläßt den »Animator«. Das Programm fragt sicherheitshalber nach, ob Sie sich auch sicher sind.

3.18 Tips zur Handhabung der Kamera

Bereits in Kapitel 3.11 wurde erwähnt, daß »Reflections-Animator« das Konzept der Kamera-Sequenzen kennt. Sie dienen zur Definition von Kamerafahrten. Dabei bewegt sich während der Animation die Kamera und damit der Betrachterstandpunkt – für den Film eine Quelle zusätzlicher Dynamik.

Allerdings wird diese Dynamik auch mit einem Nachteil erkauft. Bei Kamerafahrten wird der Umfang der Animationsdatei sehr viel größer, wenn man nicht auf einige wichtige Details achtet. Kapitel 4 enthält einige Tips zum sparsamen Speicherverbrauch bei Kamerafahrten.

Eine Kamera-Sequenz ist aber auch bei feststehender Kamera nötig. Denn der »Animator« holt sich alle Angaben über Objekte aus Sequenzen. Und für ihn ist die Kamera nichts weiter als eines der Objekte.

3.18.1 Feststehende Kamera

Auch wenn die Kamera während der ganzen Animation nicht ein einziges Mal ihren Standpunkt ändert, muß eine Sequenz für sie angelegt werden (mit dem Menüpunkt »neu_seq.kamera«). In diesem Fall genügt es allerdings, die Sequenz mit einem einzigen »Snap« für den aktuellen Kamerastandort auszustatten. Ist in einer Sequenz nur ein »Snap« enthalten, so gelten dessen Werte für den gesamten Zeitraum der Animation als konstant. Sind mehrere »Snaps« gesetzt, so müssen sie die komplette Dauer der Animation abdecken, da sonst die Kameraposition außerhalb der »Snaps« nicht definiert ist.

3.18.2 Bewegte Kamera

Sicherlich macht diese Sorte Effekte einen Film interessanter. Eine Animation, die den Betrachter durch die Szene fährt oder fliegt, wirkt schon viel interessanter und lebendiger. Dennoch sollte man sparsam damit umgehen, um den Zuschauer nicht ständig damit zu konfrontieren.

Wichtig ist aber hierbei ganz besonders die richtige Wahl der Sequenz-Flags. Die ersten drei Flags (»hüpf«, »linear« und »weich«) bestimmen, wie der Bewegungspfad aussieht, während die letzten beiden (»Richtung« und »Ziel«) festlegen, wohin die Kamera ihr Objektiv richtet.

Wenn Sie weder das Richtungs- noch das Ziel-»Flag« setzen, so nehmen Sie die Kamerasteuerung selbst in die Hand. Die Richtung wird dabei über die Kamera-Regler eingestellt. An markanten Bewegungs-Eck-Punkten sollten »Snaps« gesetzt werden. Die Bewegung und Blickrichtung der Kamera dazwischen interpoliert der »Animator« wie gewohnt selbständig.

Noch interessanter wirkt es aber, wenn die Kamera auf ein festes Ziel, einen der an der Szene beteiligten Körper gerichtet ist. Dann genügt es, das Zielflag zu setzen und den Körper anzugeben, auf den die Kamera zielen soll. Wohin auch immer sich Kamera und Zielkörper von jetzt an bewegen, die Kamera zielt stets auf den Mittelpunkt des Körpers. Eventuell vorhandene Richtungsangaben für die Kamera in den »Snaps« werden dabei ignoriert. Allerdings hat der Anwender weiterhin vollen Einfluß auf den Roll-Winkel und das Teleobjektiv.

Die zweite Möglichkeit, die Richtung der Kamera zu bestimmen, ist das Richtungsflag. Ist dieses gesetzt, so richtet sich die Kamera immer in Flugrichtung aus – mit Blick entweder stur nach vorne oder nach hinten.

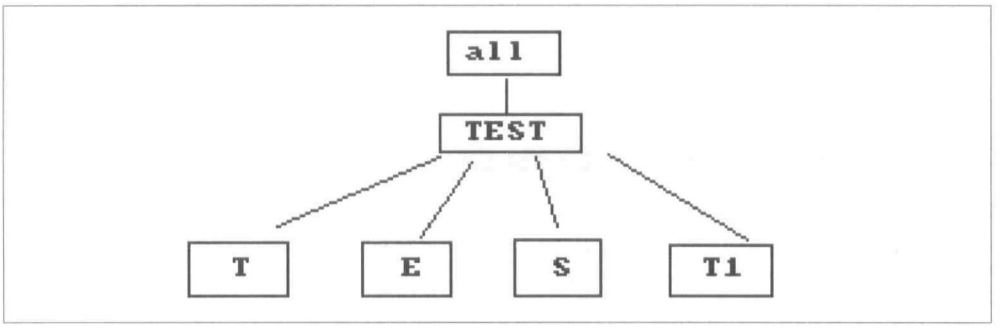


Bild 24: Der Körperbaum für die Originalszene

3.19 Körperbaum und Sequenzen

Der Körperbaum ist für »Construct«-Benutzer zunächst relativ unwichtig und daher etwas ungewohnt. Die Vor- und Nachteile werden erst mit zunehmender »Animator«-Erfahrung klar. Anfangs ist es am besten, wenn Sie sich nicht groß um den Körperbaum kümmern. Definieren Sie zunächst alle Körper direkt als Söhne des Körpers »all«. Im Falle der Testszene (die Datei »Test« auf der Programmdiskette) bedeutet das, daß die vier einzelnen Buchstaben voneinander völlig unabhängig sind. Die Original-Szene enthält zusätzlich einen weiteren Körper »Test«, der die vier Buchstaben zusammenfaßt. Bild 24 stellt den daraus resultierenden Körperbaum dar.

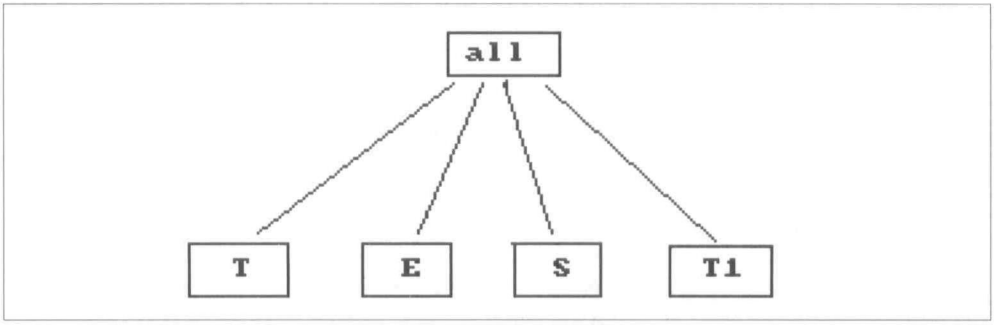


Bild 25: Der Körperbaum mit gelöschtem »Test«-Körper.

Löscht man dagegen vorher in »Construct« den Körper »Test« mit der Funktion »k_löschen«, so würde vom »Animator« der in Bild 25 gezeigte Baum erzeugt werden.

Beide Konstellationen unterscheiden sich auf den ersten Blick kaum. Bei näherer Betrachtung sind die Unterschiede jedoch gravierend. Im zweiten Beispiel existiert kein Körper »Test« mehr. Aus diesem Grund ist es nicht mehr so einfach, die vier Buchstaben gleichzeitig zu bewegen oder zu drehen, denn der Körper »all« läßt sich

nicht mehr beeinflussen. Einzeln bewegen lassen sich die Buchstaben natürlich. Dann braucht aber jeder der vier seine eigene Sequenz. Für eine Animation, bei der die einzelnen Buchstaben unabhängig heranfliegen, um schließlich den Schriftzug »Test« zu bilden, läßt sich das gut machen. Falls der Schriftzug »Test« als Ganzes behandelt werden soll, wird das ohne den Körper »Test« umständlich.

Bild 26: Eine Bewegung des Gesamtkörpers ...

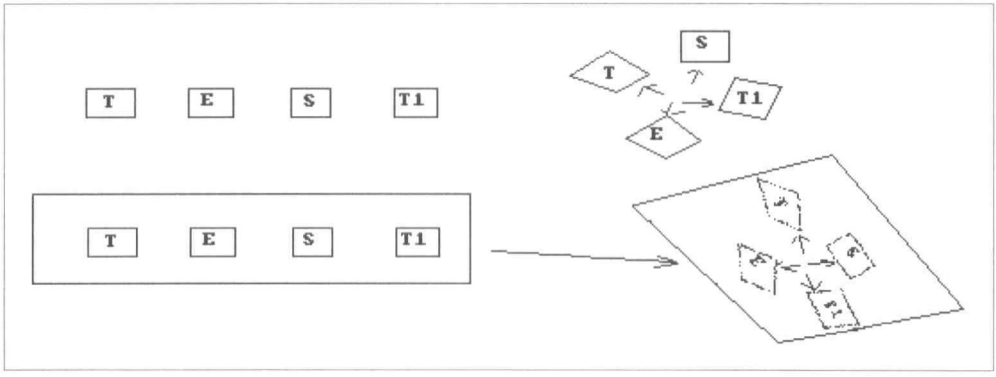


Bild 27: ... und seiner Einzelteile

Um den Körper »Test« als Ganzes zu bewegen, ist es am einfachsten, die Konstellation aus Bild 24 zu verwenden. Der Anwender braucht dann nur eine Sequenz für »Test« und kann dann alle Bewegungen und Drehungen damit durchführen. In diesem Fall könnte man auch auf die Körper »T«, »E«, »S« und »T1« verzichten.

Anders sieht es dagegen aus, wenn der Körper »Test« nicht nur als Ganzes, sondern auch als einzelne Buchstaben bewegt werden soll. Dann werden Sequenzen für das Ganze als auch für seine Einzelteile gebraucht. Und genau an diesem Punkt wird zum erstenmal die Baumstruktur richtig interessant. Die Grundlagen hierzu werden in den Bildern 26 und 27 grafisch verdeutlicht.

Um die Bewegung von »Test« mit der seiner Buchstaben zu kombinieren, genügt die Funktion »multi_seq« aus dem Sequenzmenü. Die Koordinaten der Körper »T«, »E«, »S« und »T1« beziehen sich auf das Koordinatensystem von »Test«. Daher werden ihre Einzelbewegungen automatisch mit der Gesamtbewegung kombiniert.

Die Kombination der Sequenzen von Körpern mit Sequenzen ihres Vaters hat viele Vorteile. Sie bietet sich besonders bei »frei-fliegenden« Körpern an. Definieren Sie zuerst die Bewegungen der Einzelkörper. Erst wenn diese zufriedenstellend ist, kommt die Bewegung des Vaters dran. Das Test-Beispiel läßt sich wunderbar auf Logo-Animationen übertragen, wie sie bei den abendlichen Fernsehshows inzwischen zum Standard geworden sind.

Weitere Beispiele für Sequenz-Kombinationen:

■ Ein fliegender Vogel. Definieren Sie zuerst die Bewegungen der Einzelteile des Vogels (Auf- und Abschlagen der Flügel), während der Vogel in Grundhaltung ist. Anschließend definieren Sie die Flugbahn des Vogels. Der Vogel wird dann flügelschlagend seine Bahn entlang fliegen.

■ Ein Flugzeug. Sie haben ein Flugzeug mit folgenden Körpern: Rumpf, Tür, Propeller, Flugzeug. Flugzeug ist dabei der Vater von Rumpf, Tür und Propeller. Zuerst kommt die Definition der Propellerdrehung dran. Während des Fluges soll vielleicht die Tür aufgehen. Definieren Sie also das Aufgehen der Tür. Alles noch in Ruhelage des Flugzeugs. Erst danach wird der Flugzeug-Körper bewegt. Lassen Sie ihn einen Looping drehen, eine Kurve fliegen.

Es gibt aber auch Anwendungen, in denen diese Kombinationen nicht hilfreich, sondern eher störend und verwirrend sind. Um dies zu vermeiden, sollte man sich klarmachen, welche Sequenz die Hauptbewegung enthält, bevor man beginnt die Sequenzen zu definieren. Im Beispiel mit dem Schriftzug »Test« führen die Sequenzen der Einzelbuchstaben kleine Bewegungen und Drehungen aus, der Schriftzug als Ganzes wird aber von der Sequenz »Test« bewegt. Bei dem Vogel ist es ähnlich. Die Einzelteile führen sekundäre Bewegungen aus, während die Hauptbewegung, die Flugbahn, vom Körper Vogel ausgeführt wird. Anders sieht es bei einem Roboter aus.

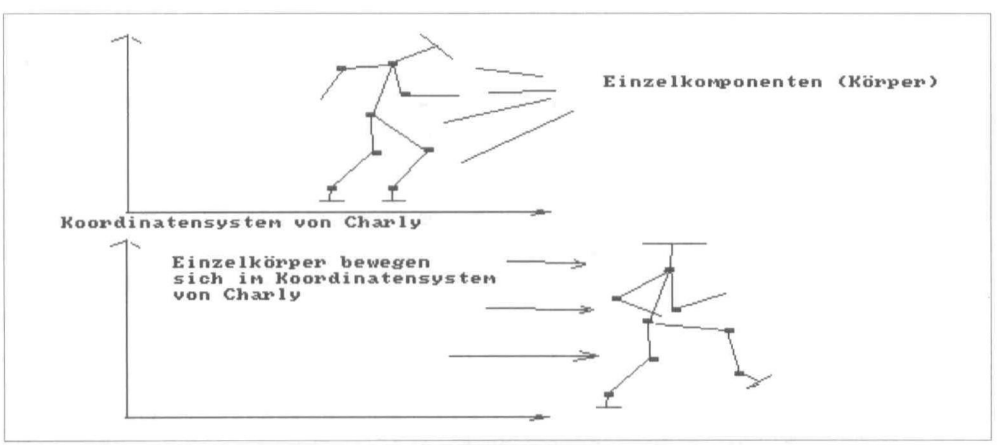


Bild 28: Das Koordinatensystem von »Charly«, wenn seine Komponenten bewegt werden

Die Teilkomponenten des Roboters sind besser über Gelenke verbunden. Ihr Vater ist beispielsweise der Gesamtkörper »Robot«. Wenn der Roboter Schritte ausführen soll, so bewegt er seine Teilkomponenten, wie »rechter_Fuß«, »linker_Fuß«, »Becken«, »Kopf« etc.

Der Gesamtkörper »Charly« hingegen bewegt sich vorerst nicht. Auf den ersten Blick widerspricht das den zuletzt genannten Regeln. Aber: Die Einzelkomponenten bewegen sich innerhalb des Koordinatensystems ihres Vaters (»Charly«) weiter. Das Koordinatensystem von »Charly« jedoch bleibt unverändert (siehe Bild 28).

Probleme tauchen allerdings auf, wenn man den Roboter außerdem noch als Ganzes bewegen möchte. Denn dann überlagern sich die Bewegungen der Einzelkomponenten mit denen des Gesamtkörpers. Die Schrittbewegungen sind in der Regel so abgestimmt, daß die Füße genau auf dem Fußboden positioniert sind, der eine Fuß bleibt stehen, während sich der andere weiterbewegt etc. Dies würde durch eine zusätzliche Bewegung von »Robot« mächtig gestört werden. Sollte dieses Problem auftauchen, ist es mit einiger Handarbeit natürlich dennoch lösbar. Man sollte aber eine zusätzliche Sequenz mit »Robot« als Komplettsystem, wann immer dies möglich ist, vermeiden.

3.20 Sequenzen: Fragen und Antworten

Die Arbeit mit Sequenzen ist unter Umständen relativ knifflig. Aus diesem Grund sind in diesem Kapitel vier typische Fragen zusammengestellt, die nach unserer Erfahrung in der Praxis auftauchen könnten.

1. Gelten einmal definierte Gelenke zwischen Körpern für die komplette Animation?

Nein. Gelenke sind nicht an die Körperstruktur gekoppelt, sondern an die Sequenz, für die sie definiert wurden. Daher sind sie nur solange gültig, wie die Sequenz aktiv ist.

2. Kann ein Körper in mehreren Sequenzen benutzt werden?

Ja. Körper können so oft in einer neuen Sequenz verwendet werden, wie der Anwender dies wünscht. Denn Sequenzen haben einen klar definierten Gültigkeitsbereich, der vom Zeitpunkt des ersten bis zum Zeitpunkt des letzten »Snaps« reicht. Enthalten zwei Sequenzen denselben Körper, und überlappen sich Ihre Gültigkeitsbereiche nicht, dann gibt es keine Probleme, denn die Sequenzen werden nacheinander abgearbeitet.

3. Was passiert, wenn sich die Gültigkeitsbereiche zweier Sequenzen überlappen?

Dann wird es interessant. Die Frage könnte auftauchen, wenn ein »Take« definiert ist oder im »multi_seq«-Modus gearbeitet wird (bei dem mehrere Sequenzen gleichzeitig interpoliert werden) – und natürlich bei der Animationsberechnung, bei der stets

alle Sequenzen bearbeitet werden. Hier ist es zunächst wichtig, drei verschiedene Fälle zu unterscheiden.

Fall A: Beide Sequenzen enthalten keinen gemeinsamen Körper.

Dieser Fall ist völlig unproblematisch. Jede Sequenz interpoliert ihre eigenen Körper. Die andere Sequenz ist davon nicht betroffen.

Fall B: Einige Körper aus Sequenz 1 sind auch in Sequenz 2 enthalten.

Dieser Fall dürfte anfangs für einige Verwirrung sorgen. Allerdings weniger beim Programm als mehr beim Benutzer. Der »Animator« interpoliert die Sequenzen der Reihe nach, entweder so wie sie in der Sequenzdatei liegen, oder so wie der Benutzer sie in einem »Take« eingegeben hat. Im Fall der überlappenden Körper gewinnt die Sequenz, die an letzter Stelle steht.

BEISPIEL: Eine Animation enthält zwei Sequenzen für den Körper »Robot«. Beide haben den Gültigkeitszeitraum von »0.0« bis »3.0«. In der ersten geht »Robot« nach links, in der zweiten nach rechts. Wird die erste Sequenz nach der zweiten berechnet, so geht »Robot« nach links, im andern Fall nach rechts.

TIP: Unsaubere Sequenzanordnungen sind zwar dank des flexiblen Konzepts des »Animator« durchaus möglich, sie verwirren aber mehr als sie helfen und sollten daher besser vermieden werden. Dies geht am einfachsten mit Hilfe der beiden Gadgets »st« und »en«, die eine Sequenz erst beginnen lassen, wenn die letzte beendet ist.

Fall C: Eine Sequenz enthält den Vater der Körper der anderen Sequenz.

In dem Fall überlagern sich die Bewegungen, die in beiden Sequenzen definiert sind, wie weiter oben beschrieben.

4. Was passiert mit Körpern, die in keiner Sequenz enthalten sind?

Nichts. Sie behalten ihre Lage aus der Grunddatei.

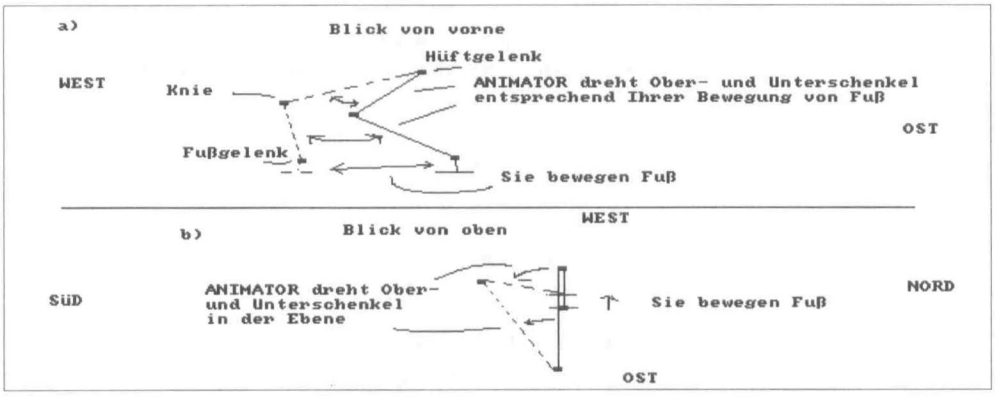


Bild 29: Das Darstellungskonzept führt zu Problemen mit Gelenken

3.21 Beachtenswertes zu Gelenken

Das Bewegen von Körpern findet beim »Animator« zweidimensional statt. Mit Hilfe der Blickrichtung ist eine Ebene arretiert, in den beiden anderen können die Körper verschoben oder gedreht werden. Dieses Konzept ist von »Construct« her bekannt und wird vom »Animator« auf die gleiche Weise verwendet. Dieses Konzept wirft aber besonders im Zusammenhang mit Gelenken auch einige Probleme auf, die erst nach einigem Probieren auffallen. Bild 29 zeigt ein solches Beispiel.

Im Bild sind Unterschenkel, Oberschenkel und Fuß über Gelenke verbunden.

a) zeigt den Blick von vorne. Wenn der Fuß mit der Maus bewegt wird, drehen sich Unterschenkel und Oberschenkel automatisch mit, so wie es vermutlich der Absicht des Anwenders entspricht.

b) zeigt den Blick von oben. Auch aus dieser Perspektive läßt sich der Fuß bewegen. Doch da der »Animator« die Unter- und Oberschenkel nur in der Nord-Süd-Ost-West-Ebene drehen, und daher das Knie nicht in der Höhe verändern kann, wird die Vorwärtsbewegung des Fußes durch Linksdrehen der Schenkel kompensiert. Das kann beabsichtigt sein, wird den Anwender aber zu Beginn doch zumindest verwundern. Auf diese Weise kann man keinen Schritt für den Roboter definieren (höchstens für einen betrunkenen Roboter).

Deshalb sollte man besonders bei Gelenkkörpern darauf achten, von wo sie gerade betrachtet werden. Ein Schritt oder Gang läßt sich am besten von der Seite definieren. Überlegen Sie sich, in welcher Ebene die Gelenke rotieren sollen (prinzipiell können sie in jeder Ebene drehen). Sorgen Sie dafür, daß die Bildschirmebene dann der Drehebene der Gelenke entspricht. Soll der Roboter von West nach Ost oder umgekehrt gehen, schalten Sie Blickrichtung vorne ein. Geht er von Nord nach Süd, dann Blickrichtung rechts oder links.

Jetzt fragt sich natürlich, was man machen soll, wenn der Roboter diagonal gehen soll. Damit ist die Drehebene der Gelenke auch schräg im Raum, entspricht jedenfalls keiner der möglichen Blickrichtungen. Dieser Situation kann man auf verschiedene Arten begegnen.

■ Sie haben nicht die Einzelkomponenten der Roboter gedreht (über »drehe_starr«), sondern den Körper »Robot«, den Vater der Komponenten. In dem Fall brauchen Sie nur den Bildschirm auf das Koordinatensystem von »Robot« auszurichten. Das geschieht mit der Funktion »körper.ausrichten«. Danach ist »Robot« nicht mehr schräg auf dem Bildschirm, sondern wieder gerade, so als wäre er in Ost-West-Richtung ausgerichtet. Jetzt können Sie die Einzelkörper von »Robot« bewegen, so wie Sie es vorhatten, und sich auf die richtige Drehebene einstellen (siehe Bild 30).

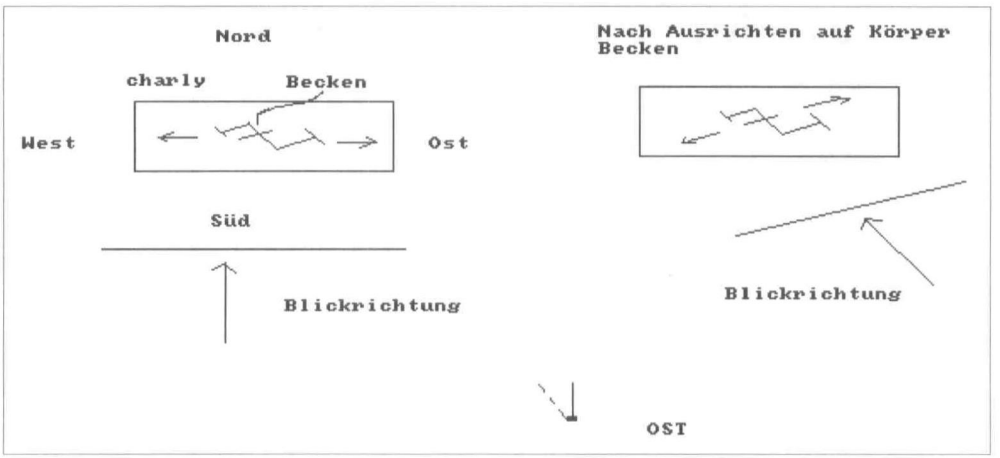


Bild 30: Der Roboter nach dem Ausrichten auf die neue Ebene

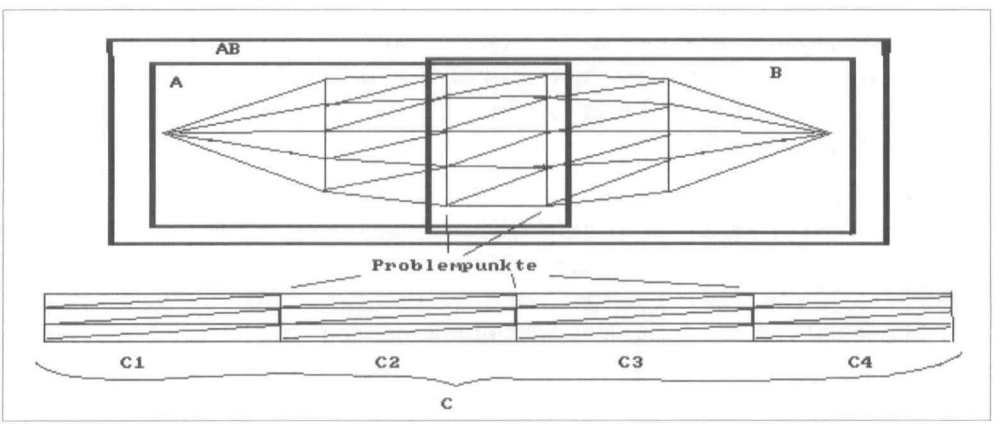


Bild 31: Beim Drehen von Komponenten muß auf diese ausgerichtet werden

■ Nicht der Vater (»Robot«) sondern die Einzelkomponenten wurden mit »Drehe_starr« rotiert. Hier nützt ausrichten auf »Robot« nichts. Sie müssen sich auf eine der Einzelkomponenten ausrichten, beispielsweise auf einen der Füße. Auch danach erscheint der Roboter wieder gerade. Der Bildschirm ist jetzt auf das Koordinatensystem von Becken eingestellt und da Becken sich starr mit den anderen Körpern mitgedreht hat, ist der Bildschirm auch im Gleichklang mit den anderen Körpern (siehe Bild 31). Sie sind damit in der gleichen Situation, als hätten Sie den Vater gedreht.

Die erste Möglichkeit ist etwas umständlicher zu handhaben, denn es müssen zwei Sequenzen statt einer bearbeitet werden. Eine für die Einzelkomponenten (die ja über Gelenke verbunden sind) und eine für ihren Vater »Robot«. Andererseits ist man hier

flexibler, was die Positionierung und Orientierung von »Robot« betrifft. Man kann »Robot« wirklich beliebig im Raum orientieren und drehen. Wenn man sich auf ihn ausrichtet, können seine Einzelkomponenten immer exakt gedreht werden.

Die zweite Möglichkeit ist normalerweise einfacher zu handhaben, dafür ist sie weniger flexibel. Solange Sie in der Ebene bleiben, gibt es wenig Probleme. Wenn Sie mit dem Roboter oder irgendeinem anderen Gelenkkörper Bewegungen definieren, und beschließen, ihn jetzt diagonal zu bewegen, so ist es in jedem Fall einfacher, die Einzelkomponenten mit »drehe_starr« in die richtige Richtung zu drehen. Wenn Sie aber den Roboter in mehr als einer Achse drehen, kann es vorkommen, daß die Einzelkörper nicht mehr so glatt interpoliert werden. Vielmehr werden sie anfangen zu »eiern«. Soll der Roboter sich also total schräg im Raum bewegen, sollte man in Betracht ziehen, die Bewegung trotz der damit verbundenen Schwierigkeiten vom Vater ausführen zu lassen.

3.22 Verzerrungen und Verbiegungen

Verzerrungen können erwünscht sein, sie sind zu diesem Zweck sogar in einem eigenen Menü, dem Körpermenü, implementiert. Eine andere Sache ist es mit Verzerrungen, die zwangsläufig entstehen, wenn Objekte (Dreiecke, Kugeln) zu mehreren Körpern gehören. Der »Animator« nutzt die Tatsache, daß ein Objekt zu mehreren Körpern gleichzeitig gehören darf, um seine Baumstruktur zu erzeugen. Die Objekte, die bei einem Roboter zum Körper »Fuß« gehören, sind ja gleichzeitig immer noch Teil des Roboter.

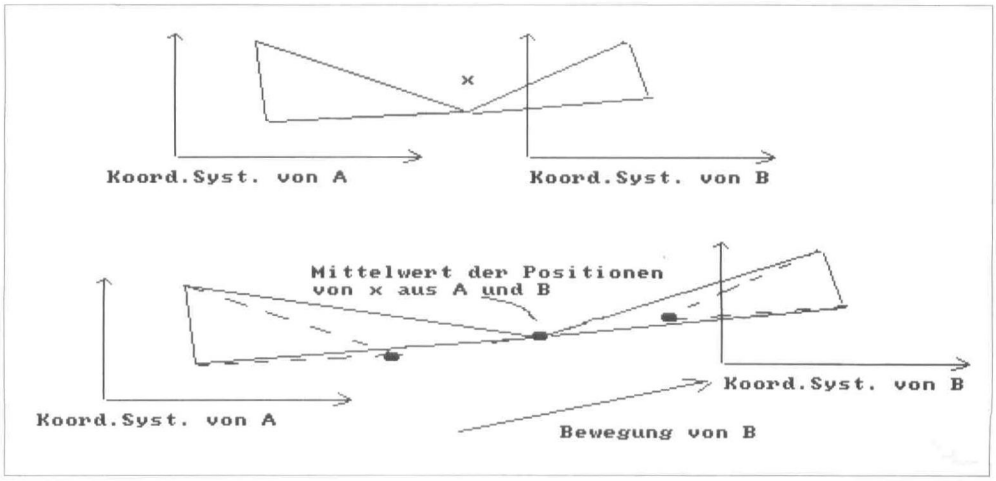


Bild 32: Gemeinsamkeiten zwischen den Objekten

Diese Fälle, bei denen Körper gemeinsame Objekte (und damit Punkte) mit anderen Brüdern haben, können aber auch Probleme verursachen. In Bild 32 sind zwei Beispiele dafür. Der Körper AB besteht aus den Teilkörpern »A« und »B«. Die beiden teilen sich einige Dreiecke gemeinsam. Körper »C« besteht aus den Teilkörpern »C1«, »C2«, »C3« und »C4«. Die Teilkörper überschneiden sich zwar nicht bei den Objekten, dafür aber bei den Punkten an der jeweiligen Grenze. D.h. einige Punkte von Objekten von »C1« sind auch Teil von Punkten aus »C2«, genauso für »C2« und »C3« sowie für »C3« und »C4«.

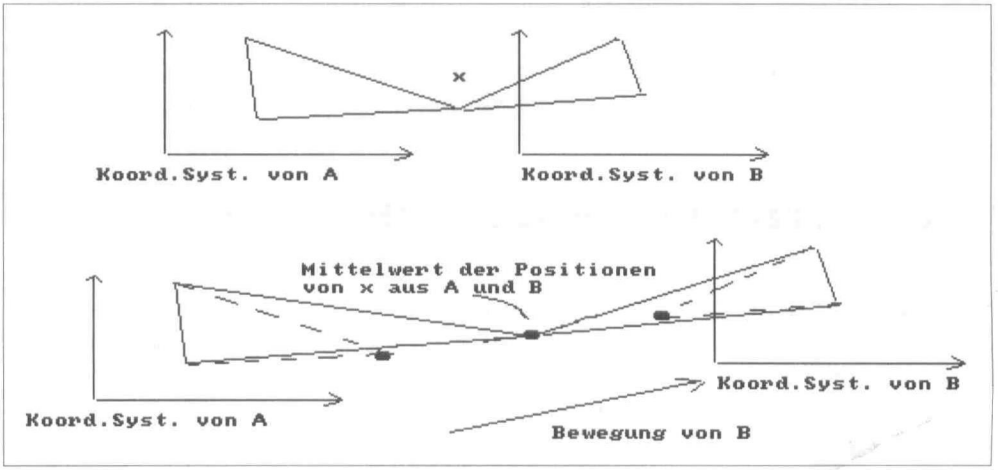


Bild 33: Bewegungen bei zusammenhängenden Körpern

Diese Fälle dürften normalerweise nicht erlaubt sein, denn in welchem Koordinatensystem sollen die Koordinaten der betreffenden Punkte berechnet werden? Da die Körper aber Brüder sind, macht das Programm hier eine Ausnahme. Der Kompromiß besteht darin, daß die Koordinaten anteilsmäßig aus allen Koordinatensystemen berechnet werden, zu denen der Punkt gehört.

Gehört ein Punkt x zu den Körpern »A« und »B«, dann werden seine Koordinaten zuerst in »A« berechnet, dann in »B«, und aus beiden wird dann der Mittelwert gebildet. Somit haben die Koordinatensysteme von »A« und »B« zu je 50% Einfluß auf den Punkt x. Diese Besonderheit zeigt sich recht anschaulich, wenn sich einer der beiden Körper bewegt. Die Bewegung wirkt sich dann auch nur zu 50% auf Punkt x aus (siehe Bild 33).

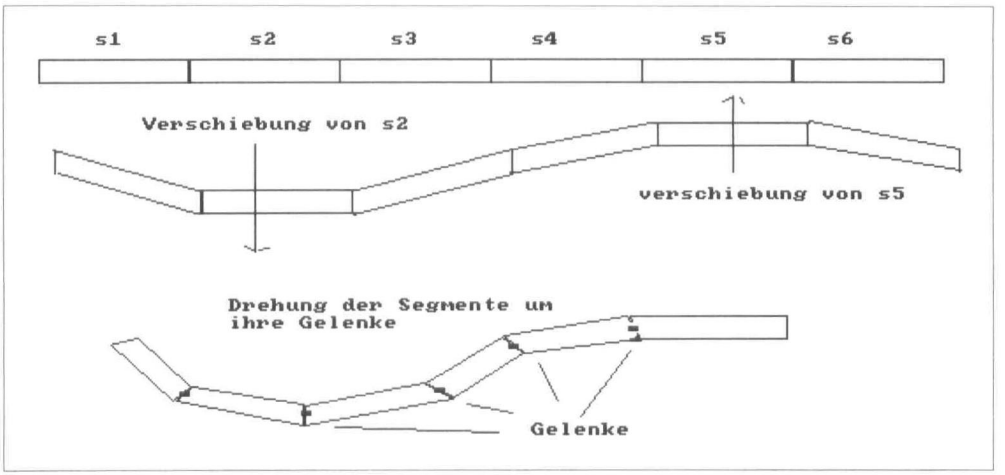


Bild 34: Ein Schlauch aus zusammenhängenden Segmenten

Analog geschieht das Ganze, wenn ein Punkt zu drei oder noch mehr Körpern gehört. Auch dann müssen aber die Körper, zu denen die betreffenden Punkte gehören, Brüder sein, d.h. sie müssen den gleichen Vater haben.

Der Nutzen dieser Techniken läßt sich erst mit etwas Übung voll ausreizen. Denn dadurch sind viele ungewöhnliche Arten von Verzerrungen möglich. Ein Beispiel wäre etwa ein biegsamer Schlauch. Der läßt sich aus mehreren Segmenten eines Rotationskörpers erzeugen (siehe Bild 34). Entweder man verschiebt einzelne Segmente, oder man baut zwischen den einzelnen Segmenten Gelenke ein. Letzteres wurde auch für den biegsamen Lampenhals vom Titelbild verwendet.

3.23. Tips und Tricks mit Sequenzen

In diesem Kapitel werden einige Tips und Anregungen angesprochen, die zu spezieller Natur sind, um sie direkt bei der Erklärung der Funktionen zu nennen. Die Tips werden in Form von Beispielen erläutert, wie sie in Animationen vorkommen können.

3.23.1. Plötzliches Auftauchen oder Verschwinden

Diese Aufgabenstellung wird vermutlich oft auftreten, wenn man seine Animationen mit einer gewissen Dramatik würzen möchte. Hier bezieht sich Auftauchen und Verschwinden allerdings immer nur auf das Gesichtsfeld des Betrachters, also der Kameraposition. Ein verschwindender Körper muß somit plötzlich für den Betrachter unsichtbar werden, beispielsweise indem er hinter die Kamera bewegt wird oder

hinter einen größeren Körper. Bei der Verwendung von »Beams« zur Bildberechnung sollte man jedoch besonders auf Spiegel achten. Den »Animator« prinzipiell auf zwei Arten Körper erscheinen und verschwinden lassen:

- Entweder indem das Hüpf-»Flag« der zugehörigen Sequenz gesetzt wird – in diesem Fall hüpft der Körper von »Snap« zu »Snap« und bleibt an der Position des »Snaps« so lange, bis der nächste kommt. Ein Auftauchen und Verschwinden ginge folgendermaßen:
- Setzen eines »Snaps« mit dem Körper außerhalb des Gesichtsfeldes.
- Setzen eines »Snaps« mit dem Körper innerhalb des Gesichtsfeldes zum Auftauch-Zeitpunkt.
- Ein »Snap« außerhalb des Gesichtsfeldes zum Zeitpunkt des Verschwindens. Wenn der Körper mehrmals auftauchen soll, wiederholen Sie die Schritte 2 und 3. Ansonsten setzen Sie den »Snap« 3 noch einmal am Ende der Sequenz.

Das Hüpf-»Flag« ist für Körper geeignet, die sich nach dem Auftauchen nicht bewegen sollen, sondern an Ort und Stelle feststehen.

Die zweite Möglichkeit sind zeitlich dicht aneinander liegende »Snaps«. Ein Körper kann auch ohne das Hüpf-»Flag« springen. Dabei wird ein Sprung einfach durch eine sehr hohe Geschwindigkeit simuliert. Schiebt man zwei »Snaps« auf der Zeitachse sehr dicht aneinander, so wird der Körper die Entfernung zwischen beiden »Snaps« mit sehr hoher Geschwindigkeit zurücklegen. Für den Betrachter springt er praktisch. An einem Beispiel in Bild 35 ist das Ganze demonstriert:

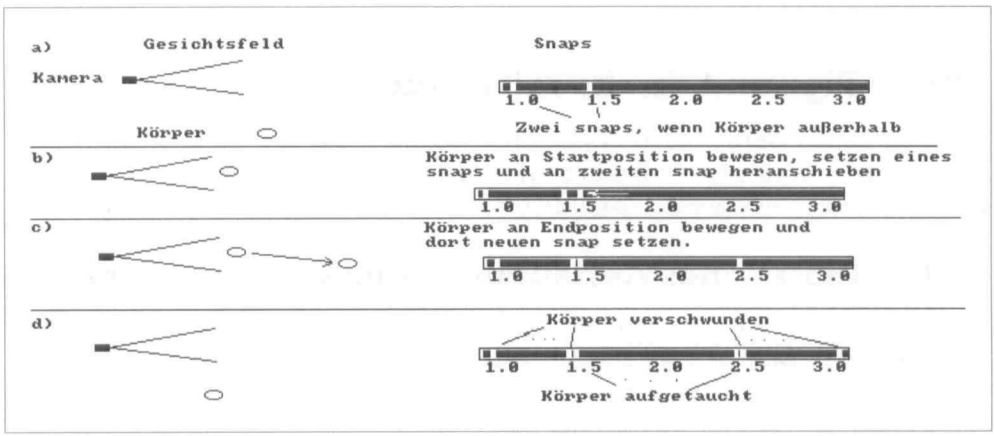


Bild 35: Auftauchen und Verschwinden von Körpern

Im Bild ist links eine Kamera mit ihrem Bildöffnungswinkel symbolisiert. Der Kreis soll einen Körper außerhalb des Gesichtsfeldes verdeutlichen. Rechts ist die Zeitskala aus

dem Sequenzfenster. Der Körper ist im Zeitraum von 1.0 bis 1.5 außerhalb des Gesichtsfeldes, deshalb sind dort zwei »Snaps« gesetzt.

Ab 1.5 soll der Körper zu sehen sein. Er wird deshalb ins Gesichtsfeld bewegt. Daraufhin wird ein »Snap« hinter 1.5 gesetzt. Um jetzt aus der Bewegung einen Sprung zu machen, wird schließlich der »Snap« nach links zum zweiten hinbewegt, bis die beiden direkt nebeneinander liegen.

Bis 2.5 soll sich der Körper innerhalb des Gesichtsfeldes bewegen. Also wird die Bewegung ausgeführt und bei 2.5 ein erneuter »Snap« gesetzt.

Ab 2.5 bis 3.0 soll der Körper wieder verschwunden sein. Folglich wird er also wieder aus dem Gesichtsfeld herausbewegt, nach 2.5 ein »Snap« gesetzt und wieder an den 2.5er herangeschoben. Ein »Snap« bei 3.0 schließt die Sache ab.

Für Sequenzen mit gesetztem »weich-Flag« beachten Sie auch noch Beispiel 3.

3.23.2 Auftauchen mit anderen Körpern

Dies ist eine Erweiterung der letzten Aufgabenstellung. Nur muß hierbei die Startposition und vielleicht auch die Endposition des Auftauchenden Körpers mit anderen Körpern synchronisiert werden. Anwendungsbeispiele wären Flugzeuge, die etwas abwerfen, Raumschiffe, die ein Shuttle freigeben oder ein Geschoß abfeuern.

Die Grundtechnik ist dabei die gleiche wie im Beispiel 1, nämlich das heraushalten des Körpers aus dem Gesichtsfeld, das plötzliche Auftauchen mittels zweier zeitlich sehr nahen »Snaps«. Nur muß jetzt der Startpunkt und die Startzeit des Körpers mit einem anderen synchronisiert werden. Als Beispiel empfiehlt sich ein Raumschiff, das ein Shuttle abwirft.

■ Zuerst wird für das Raumschiff eine Sequenz angelegt, in dem seine Bahn festgelegt ist. Das kann eine geradlinige Bahn sein oder eine Kurve, oder das Raumschiff kann rotieren, das ist egal. Schauen Sie sich die Bahn auch perspektivisch an, machen eine Skelett-Animation davon, korrigieren sie bis Sie zufrieden sind.

■ Eröffnen Sie jetzt eine neue Sequenz für das Shuttle. Bewegen Sie das Shuttle außerhalb des Gesichtsfeldes und machen Sie dann davon zwei »Snaps«, beide sollten weit in der Vergangenheit liegen.

■ Anschließend legen Sie den Zeitpunkt x fest, an dem das Shuttle vom Raumschiff freikommt. Schalten Sie wieder um auf die Raumschiff-Sequenz, bewegen Sie den Zeitcursor auf Zeitpunkt x und klicken Sie »intp« an. Damit haben Sie genau die Position des Raumschiffs zum Startzeitpunkt. Sollte es schief im Raum liegen, so ist es jetzt sinnvoll, auf das Koordinatensystem des Raumschiffes umzuschalten (mit Funktion Körper.ausrichten).

■ Schalten Sie jetzt wieder auf die Shuttle-Sequenz um. Der Zeitcursor, der in allen Sequenzen immer gleich ist, steht noch genau auf dem Zeitpunkt x. Bewegen Sie jetzt das Shuttle zu der Stelle hin, wo es vom Raumschiff aus starten soll. Und drehen Sie es so, wie es einer sinnvollen Startposition entspricht. Vergleichen Sie die Position von mehreren Seiten. Klicken Sie dann »Snap« an.

■ Schieben Sie den zweiten »Snap« der noch die »Verschwinde«-Position enthält, von links an den neuen »Snap« heran. Damit wäre das plötzliche Auftauchen des Shuttles geregelt.

■ Schalten Sie jetzt auf »multi_seq« um und definieren Sie als »take« die Raumschiff- und Shuttle-Sequenz.

■ Definieren Sie jetzt die Bahn des Shuttles mit einem oder mehreren »Snaps«, wie es ihnen sinnvoll erscheint. Da »multi_seq« gesetzt ist, können Sie mit »intp« immer Position von Raumschiff und Shuttle auf ihren Bahnen vergleichen.

3.2.3. Sprünge in weichen Kurven

Das Beispiel klingt auf den ersten Blick widersprüchlich, denn weiche Kurven zeichnen sich gerade dadurch aus, daß sie weder Knick noch Sprung haben. Doch für praktische Zwecke ist es manchmal erforderlich, daß man in eine weiche Kurve eine solche Unregelmäßigkeit einbaut.

Bild 36 zeigt eine Kurve, die durch fünf »Snaps« in ihrer Form bestimmt ist. Nur der erste und das letzte »Snap« werden von der Kurve getroffen. Man kann jetzt leicht erreichen, daß die Kurve einen der anderen »Snaps« erreicht und dort einen Knick macht. Der Trick dabei ist wieder ein Doppel-»Snap« genauer gesagt, zwei »Snaps«, die zeitlich dicht beieinander liegen.

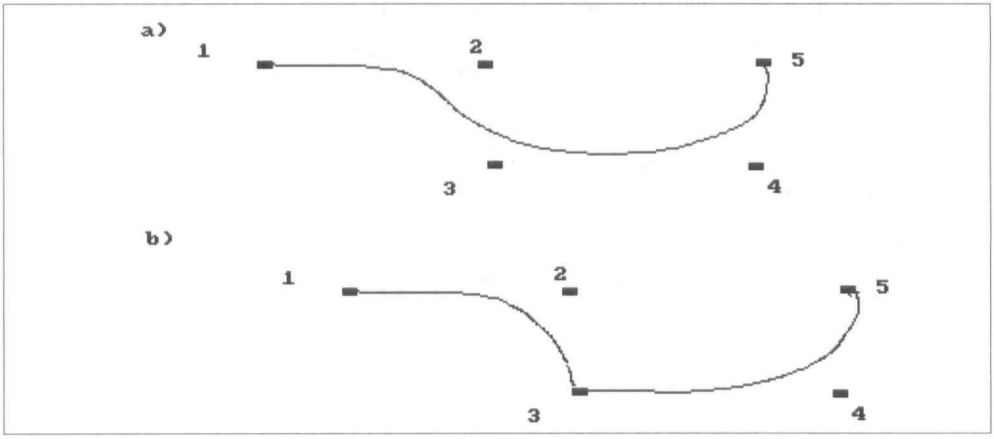


Bild 36: Fünf Snaps legen eine Kurve fest

In »Snap« 3 soll ein Knick in der Kurve entstehen. Man bewegt den Zeitcursor auf der Zeitachse zu »Snap« 3 und klickt »i2« an. Damit wird die Position direkt am »Snap« interpoliert und nicht die auf der Kurve. Der Zeitcursor wird knapp hinter »Snap« 3 bewegt. Danach wird an der Stelle ein neuer »Snap« gesetzt. Dieser neue »Snap« wird jetzt auf den »Snap« 3 zugeschoben, bis die beiden nah nebeneinander liegen. Danach macht die Kurve einen Knick, wie in Bild 36 Teil b dargestellt.

Wenn Sie bei »Snap« 3 vergrößern und mit »Sequenz.plot_pfad« die Kurve erneut anzeigen, können Sie erkennen, daß der Knick nur scheinbar einer ist. Tatsächlich besteht er aus einer sehr engen Kurve.

Bei einem Sprung sieht die Sache ähnlich aus. Nur besteht bei weichen Kurven ein Sprung aus zwei Knicken. Bild 37 zeigt diesen Sachverhalt.

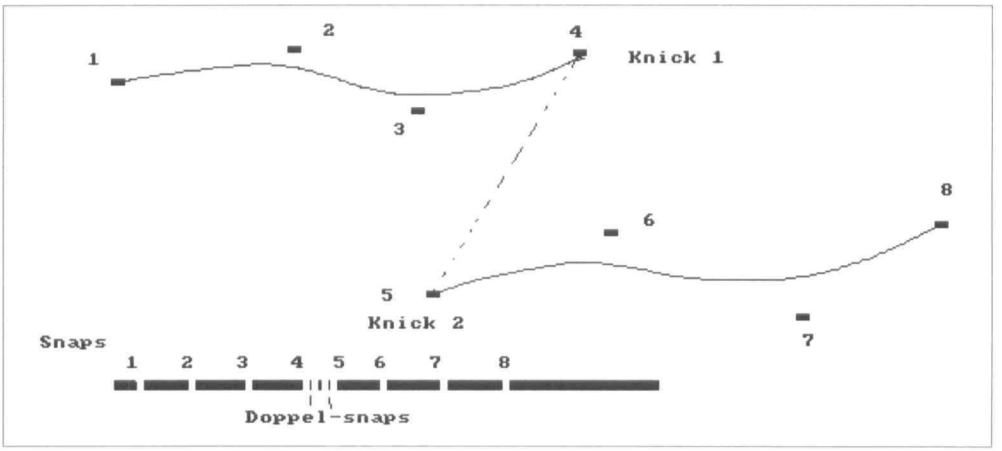


Bild 37: Ein Sprung aus zwei Knicken

Zwischen den »Snaps« 4 und 5 soll ein Sprung erfolgen. Dazu muß »Snap« 4 erreicht werden, was einen Knick an der Stelle bedeutet. Wir setzen also zwei »Snaps« an der Position von 4 dicht nebeneinander.

Da die Kurve nach »Snap« 5 normal weitergehen soll, muß auch »Snap« 5 einen Knick darstellen. Also kommen auch an die Stelle zwei »Snaps« dicht beieinander. Ein Sprung besteht also bei weichen Sequenzen aus vier nahe beieinanderliegenden »Snaps«, anstelle von zweien.

4 Make_Anim – Erzeugen von Animationen

4.1 Einleitung

Selbstverständlich ist der zentrale Sinn des »Reflections-»Animator«« das Erzeugen von Animationen. Alle bislang behandelten Themenbereiche drehten sich aber ausschließlich um den Konstruktionsteil – das Aufzeichnen von Sequenzen, Bewegungs- und Zerreffekten. Das Ergebnis dieser Bemühungen muß erst noch als farbige Animation möglichst in hochauflösender, fotorealistischer Qualität sichtbar gemacht werden. Diesen Job übernimmt »Make_Anim«. Im Programmteil »Animator« wird der Anwender wohl vergeblich nach bildbezogenen Daten suchen, so fehlten etwa so wichtige Dinge wie Bildgröße, Farbanzahl, aber auch bestimmte Raytrace-Parameter wie etwa Raytrace-Tiefe, Antialias-Stufe etc. Diese Parameter haben allerdings auch nichts verloren in einem Programm, das für die Definition von Bewegungen zuständig ist – sie würden es nur unnötig überladen und damit unübersichtlicher machen.

Diese Aufgabe übernimmt »Make_Anim«. Dieses Programm dient erstens zur Eingabe der zusätzlichen bildbezogenen Daten und dann zur unter Umständen langwierigen Durchführung der Animationsberechnung – denn Animationen erfordern von Natur aus ein aufwendiges Berechnungsverfahren. Gerade wenn dann auch noch Raytrace-Qualität angestrebt wird, ist die Rechenzeit-Skala nach oben sehr weit offen. Aufwendige Szenen können durchaus mehrere Stunden Rechenzeit in Anspruch nehmen – für ein einziges Bild. Für die Animation erhöht sich dieser Aufwand entsprechend der Anzahl der Bilder. Natürlich läßt sich der Aufwand verringern, indem man für einen einfachen Bildaufbau sorgt. Einfach heißt hier vor allem möglichst nur eine oder zwei Lichtquellen, nicht gerade alle Objekte spiegelnd oder durchsichtig, auf die »antialias«-Funktion verzichten. Bei einer Animation von 50 Bildern hat der Rechner dann eine gute Woche zu arbeiten. Wenn die Raytrace-Qualität nicht unbedingt erforderlich ist, kann man statt »Beams« auch das Programm »Scan« verwenden, was die Rechenzeit von einer Woche auf ein bis zwei Tage verkürzt.

Bevor »Make_Anim« im einzelnen erläutert wird, sollen noch die Einzelschritte erklärt werden, die das Programm bei der Animationsberechnung durchzuführt.

Eine Animation besteht aus einer Folge von Einzelbildern. Die Berechnung der Bilder wird entweder von »Beams« oder von »Scan« vorgenommen. »Beams« benötigt als Eingabe eine Szenen- und Kamera-Datei (»Scan« ebenso). Hier kommt jetzt nochmals der »Animator« ins Spiel, denn er muß für jedes Einzelbild aus der Grundszene eine

Zwischenszene berechnen, die dem Bewegungsablauf zum Zeitpunkt des Einzelbildes entspricht.

Render-Schritt: Der erste Schritt der Animationsberechnung – er wird hier der Einfachheit halber Render-Schritt (»rendern« bezeichnet den Vorgang der Bildberechnung) – besteht demnach aus zwei Teilen:

- Erzeugung der Zwischenszene durch den »Animator«
- Berechnung eines RGB-Bildes durch »Beams« oder »Scan«

Show-Schritt: In einem zweiten Schritt wird das RGB-Bild, das noch die volle Farbinformation von 16 Millionen Farben enthält, in ein IFF-Bild umgewandelt, so daß es im Amiga darstellbar ist. Dieser Schritt wird wie bei Reflections durch das Programm Show durchgeführt.

Pack-Schritt: Die Einzelbilder allein machen noch keine Animation aus. Sie werden in einem dritten Schritt zu einer Animation »zusammengepackt«. Dabei werden von Bild zu Bild nur die Änderungen gespeichert, was meistens sehr viel Speicher spart und längere Animationssequenzen überhaupt erst möglich macht. Diesen dritten Schritt erledigt das Programm »Packer«.

Die drei genannten Schritte oder Grundoperationen (RGB-Bild erzeugen, IFF-Bild erzeugen, zusammenpacken) werden für jedes Einzelbild durchgeführt. Nicht mehr benötigte Dateien werden anschließend gelöscht. So ist die Zwischenszene nach der Berechnung des RGB-Bildes überflüssig. Genauso überflüssig ist das RGB-Bild, nachdem Show das entsprechende IFF-Bild erzeugt hat, das selbst wieder gelöscht wird, sobald es mit Vorgängerbildern zusammengepackt wurde.

Im »REF:«-Ordner befindet sich neben den Icons für Reflections und Manager auch eines für »Make_Anim«. Per Doppelklick startet das Programm automatisch. Für Animationsjobs erzeugt der »Animator« ein eigenes Icon, über das ebenfalls »Make_Anim« gestartet wird. Außerdem kann »Make_Anim« über das CLI gestartet werden. Die Befehlszeile dafür lautet:

```
»Make_Anim«   <job_datei>
```

<job_datei> kann auch weggelassen werden.

4|2| Der Animations-Auftrag (Job)

Ein Animations-Auftrag oder »Job« legt die Einzelheiten der Animation fest. Diese Einzelheiten sind in einer Datei, der »Job-Datei« gespeichert. Nach dem Start von »Make_Anim« öffnet sich ein fast bildschirmfüllendes Fenster mit vielen Gadgets, sowie ein weiteres, das im Augenblick leer ist.

Die Arbeitsschritte »Render«, »Show« und »Pack« haben ihre eigenen Bereiche im Fenster und können ein- und ausgeschaltet werden. Dabei haben die einzelnen Gadgets folgende Bedeutung:

Render: Mit diesem Gadget kann der Anwender den gesamten »Render«-Schritt der Animationserzeugung ein- bzw ausschalten. Letzteres ist nur dann sinnvoll, wenn bereits eine Bildserie von RGB-Dateien vorliegt, die jetzt nur noch mit »Show« und »Pack« als Animation zusammengefaßt werden soll.

Grund-Datei: Hier wird festgelegt, welches die Objektdaten für die Animation sind. Die Objektdaten liegen in einer »Construct«-Datei. Geben Sie hier die gleiche Datei an, die Sie beim Aufruf des »Animator« genommen haben (während der Bewegungsdefinition).

Sequenz-Datei: Hier steht der Name der Sequenz-Datei, die die Bewegungspfade der Animation enthält. Der Anwender sollte dabei beachten, daß alle Sequenzen der Datei ausgeführt werden. Daher sollten im »Animator« die Sequenzen, die nicht unmittelbar zur Animation gehören, gelöscht werden.

Start-Zeit und **End-Zeit:** Wie bei den Sequenzen braucht man auch hier ein Intervall, das festlegt, von wann bis wann die Animation gehen soll. Das erste Bild der Animation wird genau die Szene zur Startzeit darstellen, das letzte Bild die Szene zur End-Zeit. Wenn Sie sich nicht bezüglich der Zeiten nicht sicher sind, starten Sie lieber noch mal den »Animator« und vergewissern Sie sich, von wann bis wann die Animation gehen soll – die Information kann aus den Sequenzen abgelesen werden. Die Zeiten sind, wie auch im »Animator« immer in Sekunden zu verstehen. »Make_Anim« setzt hier anfangs als Startzeit den Anfangszeitpunkt der am frühesten beginnenden Sequenz ein, und als Endzeit den Schlußzeitpunkt der am letzten aufhörenden Sequenz.

Bilder: Wie fein das Zeitintervall in einzelne Bilder aufgeteilt wird, legen Sie hier fest. Haben Sie die Absicht, die Bilder einzelbildweise mit einem Recorder aufzunehmen, so sollten Sie hier mit der Bildanzahl eine Bildrate von 1/25 Sekunde pro Bild festlegen. Bei einer RAM-Animation ist (leider) die Wiedergabegeschwindigkeit nicht so exakt festzulegen. Bei komplexen Animationen, bei denen sich von Bild zu Bild viel ändert, ist es oft nicht möglich, in einer 1/25-Sekunde die neuen Bilddaten aufzufrischen. 10–20 Bilder pro Sekunde sind bei RAM-Animation realistischer.

HINWEIS: Mit Anzahl_Bilder geben Sie nur an, wie fein das Zeitintervall von Startzeit bis Endzeit in Einzelbilder aufgeteilt werden soll. Welche Bilder letztendlich berechnet werden, geben Sie in den beiden danebenliegenden Gadgets »von« und »bis« an (siehe unten).

von und **bis:** Der Sinn dieser beiden Angaben leuchtet anfangs vielleicht nicht ganz ein, denn mit der Anzahl der Bilder ist doch schon genug gesagt. Das stimmt allerdings nur dann, wenn die Animation von vorne bis hinten in einem Stück durchgerechnet

wird. Bei Angabe der Anzahl Bilder wird automatisch »von« auf 0 gesetzt und »bis« auf die Anzahl Bilder minus 1. Wollen Sie eine Animation mit 40 Bildern berechnen, so geht die Bildberechnung von Bild 0 bis Bild 39. In der Regel wird aber (wegen der langen Rechenzeit) die Animationsberechnung häufig unterbrochen werden, oder Sie wollen den ganzen Animationspfad in mehrere Animationen aufteilen. Dann können Sie hier genau das Intervall eingeben, das berechnet werden soll. In Bild 38 ist dieser Sachverhalt dargestellt.

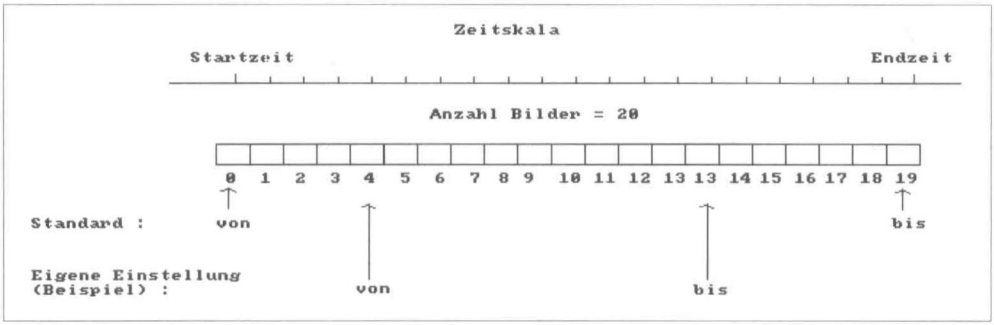


Bild 38: Ein aufgeteilter Animationspfad

Beispielsweise wäre es denkbar, eine Startzeit von 0.0 und eine Endzeit von 2.0 einzugeben, sowie eine Bildanzahl von 20. Das bedeutet eine gewünschte Abspielgeschwindigkeit von 10 Bildern pro Sekunde. Gibt man in diesem Fall nichts weiter an, werden automatisch die Bilder von 0 bis 19 berechnet. Dabei enthält Bild 0 die Objekte zum Zeitpunkt 0.0 und Bild 19 die Objekte zum Zeitpunkt 2.0. Man kann aber auch (durch Angabe in den »von« und »bis« Gadgets) ein anderes Intervall angeben, etwa von Bild 4 bis Bild 13. Dann enthält das erste Bild der Animation (Bild 4) die Objekte zum Zeitpunkt 0.42 und das letzte Bild (Bild 13) die Objekte zum Zeitpunkt 1.368.

Ray-Tie und **Raytrace-Tiefe Antial:** Antialias-Stufe

Scan und **Beams:** Dies ist praktisch ein Schalter für kurze oder lange Rechenzeit. Der verwendete Algorithmus bei der Bildberechnung kann hier eingestellt werden. Entweder wird hier »Beams« festgelegt und damit die längere Rechenzeit in Kauf genommen – dann allerdings auch mit der höheren Qualität. Oder man entscheidet sich für »Scan«, der schneller rechnet, aber nicht so gute Ergebnisse liefert. Im Fall von »Scan« sind die Angaben zu Ray-Tie und Modus ohne Bedeutung. Durch Anklicken kann zwischen »Beams« und »Scan« hin und hergeschaltet werden.

Size: Schaltet zwischen Standardbildgrößen von »LoRes0« bis »Hires4« hin und her.

Zeilen und **Spalten:** Die Bildgröße wie von »Construct« her gewohnt. Sie kann frei eingestellt werden.

Schatt: Schlagschatten

RGB-Save: Normalerweise wird die RGB-Datei, die »Beams« (oder »Scan«) liefert, gelöscht, nachdem »Show« daraus ein IFF-Bild erzeugt hat. Das RGB-Save-Gadget veranlaßt »Make_Anim«, die RGB-Datei in einen, vom Anwender bestimmten Ordner zu kopieren. Wohin, das legt das RGB-Datei-Gadget fest. Diese Option ist in zwei Fällen sinnvoll: Erstens, falls Sie eine Grafikkarte besitzen, die mehr als 4096 Farben darstellen kann. In diesem Fall kann die Bildserie einzelbildweise dargestellt und eventuell mit einem einzelbildfähigen Recorder aufgezeichnet werden. Zweitens, wenn man sich nicht festlegen möchte, mit wie vielen Farben die Animation im Amiga ablaufen soll, ob mit HAM oder mit 32 oder 16 Farben. Das ist ja auch eine Speicherfrage. Nur wenn die RGB-Dateien noch existieren, können daraus später mehrere Animationen in verschiedenen Farben erzeugt werden. Allerdings sollte in diesem Fall genügend Platz auf der Festplatte zur Verfügung stehen, denn die RGB-Dateien brauchen besonders bei Verwendung von Texturen viel Speicher.

RGB-Datei: Bei Verwendung der RGB-Save-Option, aber auch in einem anderen Fall (siehe unten) muß »Make_Anim« wissen, wohin er die RGB-Dateien abspeichern soll. Da es nicht sinnvoll ist, für jedes einzelne Bild einen eigenen Namen anzugeben, nimmt »Make_Anim« den Namen, der im RGB-Datei-Gadget steht und fügt die aktuelle Bildnummer an. Beispiel: Sie haben im RGB-Datei-Gadget den Namen »dh2:bilder/flug« angegeben. Dann erzeugt der »Animator« folgende IFF-Dateien: »dh2:bilder/flug0«, »dh2:bilder/flug1«, »dh2:bilder/flug2« usw. Falls die Bildnummer nicht am Ende des Namens stehen soll, sondern irgendwo in der Mitte, so genügt es, an der Stelle ein »*« einzufügen. »Make_Anim« erzeugt bei Angabe von »dh2:Bild*.image« die Bildserie: »dh2:Bild0.image«, »dh2:Bild1.image«, »dh2:Bild2.image« usw.

Show: Das »Show«-Gadget schaltet den kompletten »Show«-Schritt ein oder aus. Ausschalten ist dann sinnvoll, wenn nur die RGB-Datei erzeugt werden soll und keine IFF- und Animationsdatei. Aber normalerweise ist »Show« eingeschaltet. Die nächsten beiden Gadgets sind vom »Manager« bekannt.

Farben: Anzahl Farben beim IFF-Bild

Mischen: Einschalten der Misch-Option

HINWEIS: Normalerweise übernimmt »Show« die RGB-Datei, die direkt zuvor von »Beams« erzeugt wurde. Wurde »Render« ausgeschaltet, geht das ja nicht. Dann nimmt »Show« statt dessen die Datei (bzw Dateien), die im RGB-Datei-Gadget angegeben sind. Es gelten dabei die gleichen Regeln für Dateinamen, wie oben erklärt. Aus diesen Angaben kann »Make_Anim« dann eine Animation erzeugen. Das gleiche, was mit RGB-Dateien möglich ist, kann auch mit dem Ergebnis von »Show« geschehen, den IFF-Dateien.

IFF-Save: Nach Betätigen dieses Gadgets teilt »Make_Anim« mit, daß es die einzelnen IFF-Dateien aufbewahren soll. Es gelten die gleichen Regeln bezüglich Dateinamen

■ unterbrochen: Der Benutzer hat mit dem »Abbruch«-Gadget die Animation unterbrochen. Im Unterschied zum letzteren Fall (»Bild-Programm unterbrochen«) werden hier alle Vorgänge für das aktuelle Bild noch zu Ende geführt. Bis tatsächlich abgebrochen wird, kann also noch einige Zeit vergehen, je nachdem, wie weit die Bildprogramme schon gekommen sind.

■ Fertig: Der Animationsauftrag ist durchgearbeitet. Entweder können Sie mit dem »abschließen«-Gadget die Animationsdatei endgültig abschließen, oder Sie können einen weiteren Job anfügen.

■ unerklärlicher Abbruch ? ! ?: Hier ist ein Problem aufgetreten, das »Make_Anim« nicht einordnen konnte, beispielsweise ein Guru, oder vielleicht ein versehentlicher Reset. Aber das ist für »Make_Anim« meistens nicht so tragisch. Wenn Sie danach »Make_Anim« neu starten, kann zumindest bei der aktuellen Bildnummer weitergerechnet werden. Allerdings sollte die unvorhergesehene Störung nicht gerade passieren, während »Pack« in die Animationsdatei schreibt. Das wäre sehr ungesund für die betroffene Diskette oder Festplatte. Aber »Pack« schreibt immer nur während weniger Augenblicke, was die Wahrscheinlichkeit für einen Guru währenddessen relativ gering hält.

4.3 Datensicherheit und Abbrechen der Animation

Dies ist ein wichtiges Thema bei der Animationsberechnung. Kein Programm ist völlig fehlerfrei. Diese Tatsache wirkt sich bei einem Multitasking-Betriebssystem wie beim Amiga, besonders wenn kein Speicherschutz vorhanden ist, stärker aus, als bei anderen Rechnern. Bei Speicherschutz merkt die Hardware und dadurch das Betriebssystem, wenn ein Programm auf einen Speicher zugreift, der ihm nicht »gehört«. Meist kann dann das Programm abgebrochen werden, bevor es »Unfug« anrichtet. Das erfordert allerdings einen höheren Hardware-Aufwand, der meist erst bei sündhaft teuren Multi-User-Systemen (Unix beispielsweise) spendiert wird.

Beim Amiga kann nichts ein fehlerhaftes (oder bösartiges) Programm daran hindern, einem anderen Programm oder gar dem Betriebssystem den Speicher zu verwüsten. Meistens führt das zum allseits bekannten indischen Freund. Damit ist natürlich die Animationsberechnung gefährdet, denn durch die lange Rechenzeit steigt die Wahrscheinlichkeit, daß in einem der daran beteiligten Programme ein Fehler auftritt. Andererseits werden Sie auf Ihrem Amiga während der Animationsberechnung auch andere Programme laufen lassen, die sich vielleicht nicht ganz korrekt verhalten. Wie dem auch sei, Fehler sind möglich, nur sollten sie nicht eine tage- oder wochenlange Rechenarbeit zunichte machen. Dies kann die Struktur der Animationsberechnung weitgehend gewährleisten:

■ Während der Berechnung stehen die meisten Informationen zur Animation in Dateien, sind also gesichert.

■ Wenn Sie per RGB-Save oder IFF-Save die Einzelbilder abgespeichert haben, dann gibt es sowieso kein Problem, denn aus den Einzelbildern kann der bisherige Stand der Animation jederzeit rekonstruiert werden.

■ Ein anderer Fall ist das sofortige Zusammenpacken der Animation ohne Speichern der Einzelbilder. Dann steckt die gesamte Animation, die bis dahin erzeugt wurde, in der Animationsdatei. Wenn die zerstört oder beschädigt wird, ist das natürlich sehr schlecht. Haben Sie diese Datei aber auf Festplatte oder Diskette, so ist die Wahrscheinlichkeit für eine Beschädigung der Datei sehr gering. Bei jedem Bild liest und schreibt nur der »Packer« für jeweils wenige Sekunden auf die Animationsdatei, und da müßten Sie schon viel Pech haben, wenn ausgerechnet währenddessen ein Guru auftritt.

»Make_Anim« holt beim Auftreten eines Gurus alle Informationen aus der Job-Datei. Darin steht die aktuelle Bildnummer, welches Programm gerade arbeitet, der Status etc. »Make_Anim« aktualisiert die Datei ständig. Um später das Auftreten eines Gurus oder ähnliche radikale Störungen zu erkennen, scheibt »Make_Anim«, während er ordnungsgemäß arbeitet, immer einen Hinweis in die Job-Datei, daß ein unerklärlicher Abbruch erfolgt sei. Wird »Make_Anim« normal abgebrochen (über die Abbruch-Gadgets), oder hat er seinen Job erledigt, so wird dieser Hinweis wieder aus der Job-Datei gelöscht. Trat hingegen ein Guru auf, so wissen Sie später beim Neustart von »Make_Anim«, daß ein (für »Make_Anim«) unerklärlicher Abbruch stattfand, und auch, in welchem Programm er gerade arbeitete. Die Fehleranalyse wird somit erheblich erleichtert. Auf jeden Fall läßt sich die Animationsberechnung normal fortsetzen, solange nicht die Animationsdatei beschädigt wurde. Meist ist aber auch dann nur der Rechenaufwand für das aktuelle Bild verloren.

4.3.1 Abbrechen oder Unterbrechen einer Animationsberechnung

Häufiger als solche Guru-Unfälle werden normale Abbrüche vorkommen. Die Animationsberechnung soll Ihren Amiga nicht tage- und wochenlang blockieren. Nachts macht es ja meist nichts aus, aber tagsüber haben Sie vielleicht noch anderes mit dem Rechner vor. Solange genügend Speicher verfügbar ist und Sie keine gefährlichen Dinge tun, geht das schon. Die Animation rechnet dann im Hintergrund vor sich hin. Wenn aber der Speicher nicht reicht, oder Sie zwischendurch ein Computerspiel machen wollen, das kein Multitasking zuläßt, so ist es problemlos möglich, die Animation abzubrechen und später fortzusetzen. Dafür gibt es zwei Möglichkeiten:

■ Abbruch: Nachdem Sie das Abbruch-Gadget betätigt haben, wird »Make_Anim« die Animationsberechnung unterbrechen, nachdem alle Tätigkeiten für das aktuelle Bild beendet sind. Ist »Beams« noch bei der Arbeit, rechnet er noch zu Ende, dann kommen noch »Show« und der »Packer« an die Reihe. Erst dann bricht »Make_Anim« ab und verabschiedet sich. Abbruch empfiehlt sich daher dann, wenn Sie es nicht so eilig haben, oder »Beams« kurz vor seinem Ende ist, oder bereits »Show« oder »Packer« arbeiten. Der Vorteil ist hierbei, daß praktisch keine Rechenzeit verlorengeht. Beim Neustarten von »Make_Anim« wird gleich mit dem nächsten Bild weitergemacht.

■ Abbruch der Bildprogramme: Die Bildprogramme »Beams« und »Scan« beanspruchen den größten Teil der Rechenzeit. Will man die Animationsberechnung kurz unterbrechen, ist es unumgänglich, auch die Bildprogramme abbrechen zu können. Bei Verwendung des Managers gibt es ja die Möglichkeit, per Abbruch-Gadget »Beams« zu unterbrechen und später an der gleichen Stelle fortzusetzen. »Scan« öffnet beim Starten vom CLI aus ein eigenes kleines Fenster mit einem Abbruch-Gadget, das die gleiche Wirkung hat, wie der Abbruch vom Manager aus. Die Bildberechnung kann später an gleicher Stelle fortgesetzt werden.

»Make_Anim« geht beim Aufruf der Programme einen anderen Weg als Manager. Es simuliert einen Aufruf über den CLI. Sie erkennen das daran, daß alle Ausgaben der Programme in dem, anfangs leeren zweiten Fenster von »Make_Anim« erscheinen, als ob sie vom CLI aus aufgerufen worden seien.

Bei »Scan« ist es damit möglich, während der Bildberechnung abzubrechen. »Make_Anim« erkennt diesen Fall an der Existenz einer ».abbr«-Datei, die zur Bilddatei paßt. Es kopiert dann die Bild- und Abbruch-Datei in den gleichen Ordner, wo auch die Animationsdatei liegt, speichert sich einen Hinweis in der Job-Datei und verabschiedet sich.

Beim Neustart kann er dann bei der Bildberechnung an der abgebrochenen Zeile weitermachen. So geht fast keine Rechenzeit verloren (der »Animator« muß lediglich für das aktuelle Bild erneut die Zwischenszene erzeugen, aber das dauert ja nicht lange).

4.4 Kombinationen von »Render«, »Show« und »Pack«

Die drei Grundschritte bei der Animationsberechnung (»Render«, »Show« und »Pack« – wobei mit Render die Alternative zwischen »Beams« und »Scan« gemeint ist) können einzeln ein- oder ausgeschaltet werden, woraus sich verschiedene Anwendungsmöglichkeiten für »Make_Anim« ergeben.

4.4.1 »Render«, »Show« und »Pack« eingeschaltet

So dürfte die Standardeinstellung aussehen, denn auf diese Weise wird die Animation vom ersten bis zum letzten Bild berechnet und in ein Anim-File gepackt, das mit dem Programm »Play« gezeigt werden kann. Die einzelnen Arbeitsschritte sind dabei:

- Die jeweiligen Gadgets der drei Teilschritte mit den richtigen Parametern füllen (siehe Kapitel 4.3).
- Wenn genügend Speicher auf der Festplatte vorhanden ist, kann RGB-Save oder IFF-Save eingeschaltet bleiben, denn löschen kann man die Dateien immer noch. Im anderen Fall sollten diese beiden Schalter ausgeschaltet sein.
- Mit »start« die Berechnung der Animation starten.
- Nach einiger Rechenzeit hat »Make_Anim« auch das letzte Bild (dessen Nummer in »bis« eingetragen ist) bearbeitet und befindet sich im Zustand »fertig«.
- Falls keine weiteren Bilder angefügt werden sollen, genügt ein Klick auf das »finish«-Gadget. Daraufhin startet »Make_Anim« den »Packer«, der die beiden letzten Bilder der Animation mit den ersten beiden verbindet, so daß später ein zyklischer Endloslauf der Animation möglich ist.

4.4.2 »Render« eingeschaltet, »Show« und »Pack« ausgeschaltet

Diese Kombination ist sinnvoll, wenn nur die RGB-Bilder berechnet werden sollen. Normalerweise wird das nur vorkommen, wenn auch ein Ausgabegerät zur Verfügung steht, das diese Bilder aus 16 Millionen Farben darstellen kann – beispielsweise eine dafür geeignete Grafikkarte. Dafür sind folgende Arbeitsschritte notwendig:

- Gadgets im »Render«-Bereich ausfüllen.
- RGB-Save aktivieren.
- In RGB-Datei eintragen, wo die RGB-Bilder gespeichert werden sollen.
- Mit »start« Berechnung der Animation starten.

4.4.3 »Render« und »Show« eingeschaltet, »Pack« ausgeschaltet

Manche Anwender werden vielleicht nicht an der Animation als Ganzes, sondern nur an den Einzelbildern interessiert sein. Beispielsweise weil verschiedene Teile der Animation auf verschiedenen Amigas berechnet wurden. In diesem Fall müßten alle Einzelbilder erst zum Schluß zusammengesetzt werden. Bis dahin sollte keine Animationsdatei existieren. Folglich muß »Pack« ausgeschaltet bleiben. Folgende Schritte sind dafür nötig:

- Gadgets im »Render«- und »Show«-Bereich ausfüllen
- IFF-Save aktivieren.
- In IFF-Datei eintragen, wo die IFF-Bilder abgespeichert werden sollen.
- In »anim_datei« entweder den Namen eines IFF-Bildes eintragen, oder das »anim_datei«-Gadget löschen.
- Mit »start« die Berechnung der Animation starten.

4.4.4 »Show« eingeschaltet, »Render« und »Pack« ausgeschaltet

Auf diesem Weg ist es möglich, eine RGB-Bild-Serie in eine IFF-Serie umzuwandeln. Folgende Schritte sind dafür nötig:

- Bei »RGB-Datei« Name der RGB-Bildserie eintragen.
- »Show«-Gadgets ausfüllen.
- Mit »von« und »bis« bestimmen, welche Bildnummern zu bearbeiten sind.
- IFF-Save aktivieren.
- Bei IFF-Datei Name der IFF-Bildserie eintragen.
- In »anim_datei« entweder den Namen eines IFF-Bildes eintragen, oder das »anim_datei«-Gadget löschen.
- Mit dem »start«-Gadget die Berechnung der Animation starten.

4.4.5 »Pack« eingeschaltet, »Render« und »Show« ausgeschaltet

Eine Serie von IFF-Bildern muß zu einer Animationsdatei zusammengefügt werden. Das könnte beispielsweise vorkommen, wenn mehrere auf verschiedenen Computern berechnete Serien zu einer einzigen Animation zusammengefügt werden sollen (siehe Kapitel 4.4.3). Dann wäre dies der letzte Schritt. Mit folgenden Handgriffen wird er gestartet:

- In »von« und »bis« festlegen, welche Bildnummern verwendet werden.
- Die Gadgets im »Pack«-Bereich füllen.
- Bei »IFF-Datei« den Namen der IFF-Bildserie eintragen.
- Mit »start« die Berechnung der Animation starten.
- Wenn die letzte Serie angefügt ist, drücken Sie das »finish«-Gadget.

Das Zeitverhalten ist hierbei nicht festgelegt. Wer dies ändern möchte, kann mit dem Utility-Programm »anim_time« noch Einfluß darauf nehmen.

4.4.6 »Show« und »Pack« eingeschaltet, »Render« ausgeschaltet

Eine Serie von RGB-Bildern soll in eine Animationsdatei umgewandelt werden. Dazu müssen folgende Schritte befolgt werden:

- In »von« und »bis« festlegen, welche Bildnummern verwendet werden.
- Bei »RGB-Datei« den Namen der RGB-Bildserie eintragen.
- Die Gadgets im »Show«-Bereich füllen.
- Die Gadgets im »Pack«-Bereich füllen.
- Mit »start« die Berechnung der Animation starten.
- Wenn keine weiteren Bilder angefügt werden sollen, drücken Sie das »finish«-Gadget.

Für das Zeitverhalten gilt das unter 4.4.5 Gesagte.

4.4.7 Hinweise zum Speicherverbrauch

Dieses Thema ist vor allem für Anwender wichtig, die noch nicht mit einer Festplatte gesegnet sind. Im Kapitel 5 wird beim Thema »Packer« genauer auf den Speicherverbrauch einer Animationsdatei eingegangen. Wer auf Disketten angewiesen ist, tut gut daran, sich im Umfang der Animation und in der Größe der bewegten Objekte auf dem Bildschirm einzuschränken. Aber man wird wohl versuchen, gerade bei einem Trickfilm das letzte an Knalleffekten herauszuholen, und so wird es da schon Konflikte geben. Angenommen, während der Berechnung einer Animation stellen Sie fest, der Umfang der Animationsdatei wächst und wächst, was Sie beispielsweise mit Hilfe des »List«-Befehls feststellen. Außerdem zeigt der »Packer« an, wieviel Speicher er für jedes Bild benötigt hat, um es an die Animationsdatei anzufügen. Sie können absehen, daß der Platz auf der Diskette für die gesamte Animation nicht ausreichen wird. Es empfiehlt sich in dem Fall, an geeigneter Stelle mit »Abbruch1« zu stoppen und nach Einlegen einer neuen Diskette fortzufahren. Der »Packer« verteilt die Animation dann auf mehrere Disketten.

4.4.8 Weitere Hinweise

Angenommen, Sie haben »Make_Anim« aufgetragen, eine Animationsdatei zu erzeugen (per eingeschaltetem »Pack«-Gadget) und nach langer Rechenzeit ist »Make_Anim« fertig. Der Status ist auf »fertig« gesetzt. Mit »finish« können Sie nun die

Animation abschließen. Dabei erzeugt der »Packer« eine Verbindung der letzten beiden zu den ersten beiden Bildern der Animation, was beim Abspielen zu einem problemlosen zyklischen Ablauf führt.

Ebensogut könnten Sie aber auch noch weitere Bilder anfügen. Allerdings nur, solange die Animationsdatei noch nicht abgeschlossen ist, wobei Sie bei vielen Parametern völlige Freiheit haben. Es läßt sich eine andere Grund- und Sequenz-Datei benutzen, ein anderes Zeitintervall, andere Bildanzahl. Die Raytrace-Parameter lassen sich ändern. Kompressionsrate etc. Lediglich zwei Werte sind nicht mehr beeinflußbar:

- Zeilen, Spalten. Die Bildgröße in einer Animation muß durchgehend konstant bleiben.
- Farben. Auch die Farbanzahl läßt sich nicht mehr ändern. Denn die Farbtabelle der Animation ist immer die Farbtabelle Ihres ersten Bildes.

Am sinnvollsten ist das Anfügen weiterer Bilder dann, wenn sie aus der gleichen Sequenzdatei stammen. Beispiel: Sie haben einen Bewegungsablauf definiert, der von 0,0 bis 2,0 Sekunden geht. Nachdem dafür die Bilder berechnet und zusammengepackt wurden, sehen Sie, daß der Speicherplatz noch weitere Bilder zuläßt. Im »Animator« fügen Sie noch eine weitere Bewegungsdefinition an, die von 2,0 bis 3,0 reicht und sich an die schon definierte anfügt. Anschließend gehen Sie in »Make_Anim« und verändern die entsprechenden Werte:

- Startzeit auf 2,0 setzen
- Endzeit auf 3,0 setzen
- Bilder auf einen Wert setzen, der der gleichen Bildfrequenz entspricht, wie bei der bisherigen Animation. (Wenn Sie beim Zeitraum 0,0 bis 2,0 eine Serie von 20 Bildern hatten, dann geben Sie jetzt 10 Bilder an.)
- »von« auf 1 setzen (bei »von=0« würde das erste Bild das letzte Bild der bisherigen Animation überschreiben).
- »bis« auf Bilder –1 setzen, beispielsweise in diesem Fall 9.
- Alles andere unverändert lassen.
- Starten.

Es ist auch möglich, Bilder mit einer völlig verschiedenen Grund- und/oder Sequenzdatei anzufügen. Das führt dann aber beim Abspielen zu einem »Sprung« an der betreffenden Stelle – einem deutlich sichtbaren »Filmriß«. Darüber hinaus hat es auch den Nachteil, daß hierbei ein hoher Speicherbedarf zu erwarten ist. Es ändert sich ja an der Stelle oft der gesamte Bildinhalt und die Farben sehr stark, was die Möglichkeiten des »Packers«, Veränderungen zu berechnen, extrem einschränkt.

5 Der Packer

5.1 Grundlagen

Bereits im ersten Kapitel wurde der Begriff der »RAM-Animation« erwähnt. Er bezeichnet das Abspielen einer Animation direkt im Hauptspeicher (RAM) eines Computers. Denn jedes Bild einzeln von Diskette zu laden, wäre viel zu langsam, die Bewegungen dann nur noch stotternd. Eine mögliche Alternative ist das Kopieren auf ein Videoband. Dieser Weg ist jedoch ziemlich aufwendig, da hierfür ein Videorecorder benötigt wird, der die Bilder einzeln aufnehmen kann. Solche Recorder sind immer noch reichlich teuer. Doch solange die Bildfolgen komplett in den Speicher passen, ist dieser Umweg nicht nötig.

Das Problem der RAM-Animation ist der große Speicheraufwand – zumindest bei einer hohen Bildqualität. Denn ein HAM-Bild in der Auflösung 320 x 256 benötigt schon etwa 60 Kbyte an Speicher. Ohne weitere Maßnahmen wäre der Speicher schon mit wenigen Bildern voll – selbst wenn 8 Mbyte zur Verfügung stehen.

Die Lösung für das Problem ist die »Delta-Kompression«. Dieses Verfahren geht davon aus, daß sich bei der Animation von Bild zu Bild nicht allzu viel ändert. Das ist eine ziemlich realistische Annahme, wenn man bedenkt, daß auch das menschliche Auge Gelegenheit braucht, aus den Einzelbildern eine fließende Bewegung zusammenzusetzen. Folglich dürfen die Änderungen pro Bild nicht allzu sprunghaft sein.

Die Grundidee der »Delta-Kompression« ist es, nur die Änderungen eines Bildes zu seinem Vorgänger zu speichern. Je weniger sich ändert, desto weniger Speicherplatz wird verbraucht. Agiert bei einer Animation beispielsweise ein Körper vor einem unbewegten Hintergrund, so ist es durchaus typisch, daß die Delta-Daten, also die Änderungen jedes Bildes zu seinem Vorgänger nur 10 bis 30 Prozent der vollen Bildinformation betragen.

Der Benutzer sollte allerdings den Begriff der »Änderung von Bild zu Bild« genau überdenken. Der Mensch versteht darunter nämlich unter Umständen etwas anderes als der Rechner.

■ Für den Menschen steht der Gesamteindruck einer Szene im Vordergrund. Dieser ist stark an Farben orientiert. Und diese wiederum ändern sich von Bild zu Bild nur minimal.

■ Der Computer dagegen kann mit Farbeindrücken wenig anfangen. Er beachtet nur die reinen Bilddaten, was für ihn nichts anderes heißt als Zahlenwerte, die als sogenannte Bytes irgendwo in seinem Speicher abgelegt sind. Diese Zahlenwerte vergleicht er miteinander und ermittelt daraus den Unterschied zweier Bilder.

Nun könnte man vermuten, daß sich ähnliche Farben auch in ähnlichen Zahlen niederschlagen, somit also beide Sichtweisen identisch sind. Weit gefehlt. Erstens gibt es für den Computer den Begriff »ähnliche Zahlen« nicht – sie sind entweder gleich oder nicht gleich. Ähnlich bedeutet also ungleich. Zweitens bedingen oft sogar gleiche Farben wegen des etwas komplizierten Verfahrens beim HAM-Modus des Amiga ungleiche Bytes. Gleiche Farben können bei HAM auf unterschiedliche Weise gespeichert werden.

Aber ein anderer Unterschied ist noch wichtiger: Der Mensch hat beim Farbeindruck das ganze Bild auf einmal im Blick und sagt dann: »ungefähr gleich«. Der Rechner vergleicht dagegen ganz penibel Bildpunkt für Bildpunkt, und führt genau Buch über jeden Unterschied.

Richtig eindrucksvoll wird das bei einer Animation deutlich, bei der die Kamera dicht über eine Textur fliegt, beispielsweise über eine Holzfläche. Diese füllt den Bildschirm voll aus. Bei jedem Bild rutscht das Holz ein kleines Stückchen auf dem Bildschirm weiter. Der Mensch hat keine Schwierigkeiten, die Maserungen und Strukturen der Textur im Auge zu behalten, was dann zu dem – in diesem Fall gerechtfertigten – Urteil führt, der Unterschied von Bild zu Bild sei nur gering.

Der Rechner vergleicht jedoch jedes einzelne Pixel von beiden Bildern, und da sich so gut wie jeder Pixel des ersten Bildes von der gleichen Position im zweiten unterscheidet (und sei der Unterschied auch noch so gering), wird das Urteil »kaum Gemeinsamkeiten« gefällt.

Das führt zu folgenden zwei Schlußfolgerungen:

■ Damit die Delta-Kompression etwas bringt, sollte sich ein möglichst großer Teil des Bildes gegenüber der Kameraposition nicht bewegen.

■ Dies kann entweder durch eine feststehende Kamera (und damit unbewegten Hintergrund) erreicht werden, oder bei einem Kameraflug dadurch, daß ein möglichst großer Teil des Bildes aus einem einfarbigen Hintergrund besteht (beispielsweise Weltraum mit nicht zu vielen Sternen, blauer Himmel, glatte Flächen etc.).

5.2 Nachbearbeitung eines Bildes

Durch diese Gesetzmäßigkeiten beim Speicherverbrauch könnte man sich theoretisch gezwungen sehen, auf gewisse künstlerische Freiheiten zu verzichten. Beispielsweise ist eine grandiose Szenerie mit vielen Objekten und Texturen, an der ein

hochdramatischer Kameraflug vorbeiführt, von vorneherein von der Speicherzensur bedroht. Denn vernünftigerweise müßte man nach einem Blick auf die Speicheranzeige des Computers darauf verzichten. Aber der wahre Künstler läßt sich nicht von derlei technischen Banalitäten zurückhalten. Er besteht selbstverständlich auf seinem unverfälschten Film. Make_Anim erzeugt pflichtgemäß die Animationsdatei – nur um nach zwei Wochen Rechenzeit festzustellen, daß einige hundert Kbyte fehlen und der Film nur auf einem besser ausgestatteten Rechner betrachtet werden kann.

Aber es muß nicht einmal der Kameraflug über eine Textur sein, der sehr viel Speicher verbraucht. Schon bei einer relativ harmlosen Szenerie ohne Textur und feststehender Kamera können durch die Misch-Option von »Show« zwei aufeinander folgende Bilder (aus Rechnersicht) total verschieden sein. Beim Mischen überlagert »Show« nämlich die Bilder mit einem Zufallsmuster, was bewirkt, daß die wenigen Farbstufungen des Amiga (nur 16 pro Farbkomponente) nicht so deutlich auftreten – eine gute aber speicherfressende Möglichkeit, die Animation weicher erscheinen zu lassen.

Der »Packer« hat jedoch eine Möglichkeit, die Animationsdaten zwischen 10 und 50 Prozent zu verringern. Dazu müßten aber entweder die Einzelbilder (als IFF-Dateien) verfügbar sein und erneut gepackt werden, oder Sie treffen von vornherein die entsprechenden Maßnahmen.

Das Prinzip ist einfach: Das von »Show« gelieferte IFF-Bild wird innerhalb einer Toleranz so geändert, daß es in mehr Bildpunkten mit dem Vorgängerbild übereinstimmt, aus Rechnersicht dem Vorgänger also ähnlicher wird. Dies hat mit der Arbeitsweise des Programms »Packer« zu tun. Denn dieses Programm vergleicht das neue Bild und seinen Vorgänger Pixel für Pixel. Jedesmal prüft es, wie groß die Farbveränderung des Pixels im neuen Bild wäre, wenn der Wert des Pixels aus dem Vorgänger eingesetzt würde. Liegt diese Farbveränderung innerhalb einer vorgegebenen Toleranz, wird diese Änderung durchgeführt. Wenn dieser Fall oft genug eintritt, so wird sich das so veränderte Bild tatsächlich weniger vom Vorgängerbild unterscheiden als ohne diese Veränderung.

Der Nachteil ist offensichtlich, daß bei dieser Methode effektiv im Bild herumgepfuscht wird, je größer die Toleranzschwelle, desto mehr. Ist die Toleranz auf 0 gesetzt, bleibt das Bild unverändert. Bei genügend großer Toleranz hingegen würde einfach das Vorgängerbild ins neue Bild kopiert. Der »Packer« liefert dann eine überraschend kurze Animationsdatei, das Ergebnis dürfte aber nicht sehr befriedigen (es ähnelt fatal einem Standbild). Es kommt also darauf an, die Toleranz so zu wählen, daß eine gute Kompression herauskommt, aber auch keine zu großen Veränderungen an den Bildern vorgenommen werden – manchmal eine knifflige Aufgabe.

Es ist durchaus berechtigt zu fragen, ob es überhaupt sinnvoll ist, die Bilddaten nachträglich zu verändern, wo sich »Scan« oder »Beams« soviel Mühe damit gegeben

haben, oder ob die Animation durch eine, selbst geringe Toleranz nicht zu stark verfälscht wird. Dazu ist jedoch wichtig zu wissen, daß »Show« gar nicht darum herumkommt, die Originaldaten zu verfälschen. Denn sowohl »Beams« als auch »Scan« berechnen alle Bilder in der höchsten derzeit darstellbaren Qualität – mit 16 Millionen Farben. Da der Amiga – zumindest ohne zusätzliche Grafikkarte – diese Farbenpracht nicht darstellen kann, muß »Show« die Farben des Bildes durch die begrenzte Farbanzahl des Amiga möglichst gut anpassen, und dabei treten naturgemäß auch leichte Verfremdungen auf. Eine weitere Veränderung der Originaldaten durch »Show« entsteht durch die Misch-Option. »Show« gleicht die relativ geringen Farbabstufungen des Amiga (maximal 16 Stufen pro Farbkomponente) dadurch aus, daß das Bild mit einem Rauschen überzogen wird. Dieses Rauschmuster sorgt zwar ähnlich wie die Mischfarben einer Malerpalette für den Eindruck von farblichen Zwischentönen, stört aber auch extrem bei der Deltakompression und kann deshalb hier durch den »Packer« ausgeglichen werden. Dazu kommt, daß dem Betrachter beim Ablaufen der Animation nun wirklich nicht die Zeit bleibt, jedes einzelne Bild genau auf Fehler in den Bildpunkten zu untersuchen, wie das bei einem Standbild möglich wäre. Er merkt es so gut wie nicht, wenn bestimmte Pixel im Bild eine geringfügig veränderte Farbe haben.

Die Kompressionsrate und auch die Bildverfälschung hängt also stark vom eingestellten Toleranzwert ab. Aber genauso wichtig ist auch der Farbinhalt des Bildes und seines Vorgängers. Der »Packer« kann ja nur Pixel, die in beiden Bildern farbähnlich sind, gleich machen. Die Anzahl dieser »farbähnlichen« Pixel schwankt stark, je nach Szenerie, und damit auch die Kompressionsrate. Bei manchen Bildfolgen ist kaum etwas zu komprimieren (wenn sich Körper beispielsweise zu schnell über zu große Distanzen bewegen), oder wenn sich sehr kontrastreiche Texturen bewegen. Bei anderen kann dagegen oft nochmals 50 Prozent des Speicherplatzes gespart werden.

Der »Packer« bietet die Nachbearbeitung von Bildern in 7 Stufen an, die jeweils verschiedene Toleranzen widerspiegeln:

Stufe 0: Keine Nachbearbeitung. Dies ist die normale Deltakompression ohne Rücksicht auf Farbähnlichkeiten. Naturgemäß geht die auch am schnellsten.

Stufe 6: Sehr starke Nachbearbeitung

Die dazwischenliegenden Werte sind linear abgestuft. Die Stufen 5 und 6 sind extreme Veränderungen. Sie sind keinesfalls für HAM-Bilder geeignet, da diese regelrecht verwüstet würden. Bilder mit 8, 16 oder 32 Farben sind dagegen manchmal auf diese Anpassungen angewiesen. Allerdings sind die von den einzelnen Stufen verursachten Veränderungen zu sehr von der Einzelsituation abhängig, als daß man allgemein gültige Empfehlungen geben könnte. Als Standard ist am ehesten Stufe 1 bis 3 geeignet.

Die Veränderungen in den Stufen äußern sich generell darin, daß Körper einen »Schatten« hinterherziehen, einen Hauch der Position, die sie im letzten Bild einnahmen. Dieser Schatten ist je nach Stufe mehr oder weniger ausgeprägt. Wie stark er auffällt, hängt natürlich auch vom Hintergrund ab, vor dem sich der Körper bewegt.

Man kann kein Patentrezept angeben, welche Stufe nötig ist. Es hängt zu sehr von der Szene und dem persönlichen Geschmack ab. Mit den Stufen 2 oder 3 liegt man aber in Standardsituationen oft richtig. Die Nachbearbeitung kostet natürlich auch Rechenzeit. Sie liegt pro Bild ungefähr in der Größenordnung von »Show«.

Generell kann man sagen, daß sich die Nachbearbeitung dann am meisten lohnt, wenn sich großflächige Objekte langsam im Bild bewegen. Dabei bedeckt ein solches Objekt zum Großteil den gleichen Bildbereich, den es schon im Vorgängerbild innehatte. Somit ist die Wahrscheinlichkeit groß, daß der bedeckte Bereich die gleiche Farbe enthält. Dann kann »Packer« einiges herausholen.

Sind diese Objekte von Texturen bedeckt, so kann »Packer« dann etwas herholen, wenn die Textur nicht zu viele verschiedene Farben enthält. Die Standard-Holztextur aus »Reflections« ist ein gutes Beispiel dafür. Die Farben darin sind sich sehr ähnlich und unterscheiden sich größtenteils nur durch die Helligkeit. Fliegt beispielsweise die Kamera langsam über eine Fläche mit der Holztextur, so kann »Packer« bei den Stufen 3 bis 5 zwischen 10 und 50 Prozent Platz sparen. Allerdings verwischt dabei die Holztextur etwas.

Nachteilig wirkt sich die Nachbearbeitung allerdings gelegentlich aus, wenn sich das bewegende Objekt und der Hintergrund farblich nicht allzusehr unterscheiden. Dann kann es nämlich vorkommen, daß das bewegende Objekt den schon erwähnten Schatten bzw. eine »Schmutzspur« hinter sich herzieht. Ist dies bei sehr dynamischen Szenen vielleicht auch manchmal wünschenswert, so kann es doch auch gewaltig stören. Denn die dem Licht abgewandte Seite der Objekte ist oft sehr dunkel und unterscheidet sich daher für den »Packer« kaum vom Schwarz eines dunklen Hintergrunds. Daher läßt er beim nächsten Bild manchmal den dunklen Teil des Objekts stehen. Bei solchen Szenen empfiehlt es sich die Toleranz der Nachbearbeitung auf 0 oder 1 zu setzen.

5.3 Aufruf und Handhabung

Sicher die einfachste Möglichkeit für den »Packer« ist der automatische Aufruf durch »Make_Anim«. Der Anwender hat damit nichts zu tun und muß nur ein paar Parameter bei der Definition des Jobs eingeben (siehe Kapitel 4.1).

Die andere Möglichkeit ist der Aufruf des »Packers« vom CLI oder der Shell aus. Damit wird im wesentlichen das gleiche bewirkt, aber in einigen Fällen ist es unerläßlich und auch praktisch.

Angenommen, es existiert eine Reihe von IFF-Bildern, die zu einer Animation zusammengefaßt werden sollen. Es könnte ja sein, daß Sie die IFF-Bilder der zuvor erzeugten Animation aufbewahrt haben und es jetzt mit anderen Kompressionsstufen erneut versuchen wollen. Der Aufruf des »Packers« geschieht dann nach folgendem Muster:

```
»Packer« <Auftrags_nr> <Anim-Datei> [<param1> <param2>]
```

Bei den Dateinamen sollte nach Möglichkeit immer der volle Pfad angegeben werden.

<Auftrags_nr> steuert die Aktionen von »Packer«. Damit ist auch die Anzahl der weiteren Parameter festgelegt. <Auftrag> kann Werte zwischen 1 und 4 annehmen. Die verschiedenen Nummern bewirken folgendes:

1: Eröffnen einer neuen Animation. Der Aufruf muß dann so lauten:

```
»Packer« 1 <Anim-Datei> <IFF-Datei>
```

■ <Anim-Datei> ist der Name der Datei, in der die neue Animation gespeichert werden soll.

■ <IFF-Datei> ist der Name des ersten Bildes der Animation. Das sollten Sie also schon erzeugt haben, bevor Sie »Packer« aufrufen.

Wenn alles zufriedenstellend gelaufen ist, wird die Anim-Datei geschlossen und erst beim nächsten Aufruf des »Packers« wieder geöffnet und bearbeitet. So können zwischenzeitliche Gurus oder das Ausschalten des Computers der Datei nichts anhaben (es sei denn sie liegt im RAM).

2: Anhängen eines neuen Bildes an die Animation. Aufruf:

```
»Packer« 2 <Anim-Datei> <IFF-Datei> <komp-stufe> <Delay>
```

■ <Anim-Datei> ist die Datei, die zuvor (mit »Packer« 1) geöffnet wurde. An diese wird ein neues Bild angehängt.

■ <IFF-Datei> ist der Name des anzuhängenden Bildes.

■ <komp-stufe> ist die Nummer einer der obenbeschriebenen Kompressionsstufen (0 bis 6). Hier haben Sie den Vorteil die Kompressionsstufe für jedes Bild individuell eingeben zu können.

■ <Delay> ist die vorgegebene Zeitdauer, die das Bild später beim Abspielen am Bildschirm bleiben soll. <Delay> wird in vielfachen von 1/60 Sekunden angegeben. <Delay>=1 bedeutet 1/60 Sekunde, 2 = 1/30 Sekunde usw.

3: Schließen der Animation. Aufruf:

```
»Packer« 3 <Anim-Datei> <Delay>
```

■ <Anim-Datei> ist wieder der Name der Animationsdatei, die geschlossen werden soll. Der Anwender sollte wirklich sicher sein, daß die Animation auch fertig ist, denn wenn sie hiermit geschlossen wird, können nicht so einfach neue Bilder angehängt werden. »Packer« hängt jetzt noch Informationen an die Datei, um von den letzten Bildern wieder zu den ersten beiden zu springen. Damit kann die Animation zyklisch abgespielt werden.

■ <Delay> siehe oben.

4: Information über den Status. Aufruf:

```
»Packer« 4 <Anim-Datei>
```

■ <Anim-Datei> ist der Name der Datei, über die Informationen gewünscht werden.

Hier gibt »Packer« aus, wie viele Bilder die Datei enthält. Das ist nützlich, wenn man mit den Bildnummern durcheinander gekommen ist. Außerdem wird mitgeteilt, wie groß das Bildformat ist, und wie viele Bitplanes es enthält.

5.4 Abschätzen des Speicherverbrauchs

Mit dem Speicher ist es immer wieder dasselbe: Egal wieviel man hat, er reicht nie aus. Je mehr Speicher vorhanden ist, desto aufwendigere Animationen werden Sie planen. Das Speicherproblem bleibt also immer bestehen.

Es ist daher gut, wenn man schon vor der Animationserzeugung abschätzt, wie hoch ungefähr der Speicherbedarf sein wird und ob eine solche Animation überhaupt speichermäßig machbar ist. Denn der Speicherbedarf hängt von folgenden Faktoren ab:

■ Die Größe des Bilder.

■ Die Farbtiefe der Bilder. Je mehr Farben die Bilder enthalten sollen, desto mehr Speicher wird pro Bild benötigt. Ein HAM-Bild (mit 4096 Farben) braucht beispielsweise doppelt so viel Speicher, wie ein Bild mit 8 Farben.

■ Die Bildanzahl.

■ Der Animationsinhalt. Je mehr sich aufeinanderfolgende Bilder unterscheiden, desto mehr Speicher braucht die Animation.

Punkt 1 und 2 führen zu folgendem Schluß: Je kleiner das Bild und je geringer die Auflösung, desto weniger Platz braucht man pro Bild. Normalerweise möchte man auf den Bildern aber auch Einzelheiten erkennen, daher sollte die Auflösung nicht zu klein gewählt werden.

Beim Amiga ist die Bildgröße zudem noch mit der Farbtiefe gekoppelt. Der farbenreiche HAM-Modus ist nur bei LoRes-Auflösung (maximal 352 Punkte horizontal) möglich. Bei der vertikalen Auflösung kann man jetzt noch zwischen normaler und doppelter Auflösung wählen (dem sogenannten Interlace). Die doppelte Auflösung ist auch mit heftigem Flimmern verbunden, wenngleich das Flimmern bei einer Animation nicht so stark auffällt, da die Bilder normalerweise – wie beim Fernsehen – in Bewegung sind.

Neben dem zusätzlichen Speicherverbrauch für die doppelte Auflösung kommt hier noch der zusätzliche Aufwand beim Abspielen der Animation hinzu. Das führt dann leichter zum Ruckeln des Films, wenn man nicht gerade mit einer Turbokarte gesegnet ist.

Auch bei der Farbanzahl kann enorm an Speicher gespart werden. HAM ist zwar schön, aber zur Not kann es auch eine Animation mit 8, 16 oder 32 Farben sein. Wenn in Ihrer Szene nicht zu viele verschiedene Farben auf einmal zu sehen sind, kann die Animation auch so gut wirken. Ein anderer Faktor ist zudem die Dynamik der Animation. Wenn sich dort viel bewegt, wenn dem Auge viel geboten wird, fallen die Farbsprünge auch nicht so sehr auf.

Wenn Sie zudem bei »Show« auf die Mischoption verzichten, kommt das auch dem »Packer« zugute.

Punkt 3 sagt aus, daß der Platzbedarf mit der Menge an Bildern steigt. Ist dies auch keine allzu große Neuigkeit, so lohnt sich doch ein kurzer Blick darauf. Man sollte sich in jedem Fall überlegen, ob man nicht auch mit weniger Bildern auskommt. Allerdings ist hier die Sache auch nicht so einfach. Je feiner ein Zeitintervall in Einzelbilder aufgeteilt wird, desto weniger ändert sich von Bild zu Bild. Wenn eine Filmsekunde in 20 Bilder aufgeteilt wird, sind die Veränderungen von Bild zu Bild gewiß geringer, als bei der Aufteilung in 10 Bilder pro Sekunde. Bei Kameraflügen ist die Sache allerdings anders. Oben wurde bereits angeführt, daß Mensch und Rechner unterschiedliche Auffassungen von Bildunterschieden haben. So gilt denn die Rechnung von weniger Bilddifferenzen bei feinerer Bildaufteilung nur, wenn sich im ganzen Bild nicht allzuviel ändert. Einfacher gesagt, bei Kameraflügen werden Sie mit 20 Bildern pro Sekunde doppelt soviel Speicher verbrauchen wie bei 10 Bildern pro Sekunden. Dem Rechner ist es egal, ob sich pro Bild, der ganze Inhalt um einen oder um zwei Pixel verschiebt. Haben Sie allerdings eine festen Hintergrund, so ist es gut möglich, daß bei feinerer Aufteilung pro Bild sogar weniger gespeichert wird.

Den größten Einfluß hat erfahrungsgemäß der Inhalt der Animation (Punkt 4) auf den Speicherverbrauch. Indem Sie festlegen, was sich bewegt, wie groß die Objekte am Bildschirm erscheinen, wo Texturen liegen sollen, bewegt oder statisch, ob Kameraflüge stattfinden, bei welchem Hintergrund usw. Diese Dinge beeinflussen ganz

entscheidend den Speicherverbrauch. Hier gibt es auch keine festen Regeln, man kann sich bestenfalls auf seine Erfahrung und eine gute Nase verlassen. Bestenfalls kann die Regel gelten, daß es anfangs besser ist, sich etwas mit Effekten einzuschränken.

Zum Schluß noch eine kurze Sammlung von Erfahrungswerten, die bei der Speicherabschätzung helfen können:

■ Die Deltadaten pro Bild werden im ungünstigsten Fall so umfangreich, wie der Bildspeicher, das sind bei LoRes-Auflösung ohne Interlace folgende Größen: HAM ca. 60 Kbyte, 32 Farben 50 Kbyte, 16 Farben 40 Kbyte, 8 Farben 30 Kbyte.

■ Eine Animation kann maximal an Speicher verbrauchen: Anzahl Bilder x Bildspeicher-Größe.

■ Erzeugen Sie von dem Zeitraum, von dem Sie glauben, daß sich dort (vom Bildinhalt her) am meisten verändert, eine Kurzanimation von 4 bis 8 Bildern (mit »Scan«, ohne Antialias). Das dauert nicht lange und Sie können dann anhand des Umfangs der Animationsdatei abschätzen, wieviel in der ungünstigen Passage der Animation ungefähr verbraucht wird. Denn »Packer« meldet nach jedem neuen Bild, wieviel Byte an Speicher er für das neue Bild einplanen mußte. Das gleiche können Sie auch für eine Passage wiederholen, in der sich nach Ihrer Meinung wenig ändert. Mit beiden Werten können Sie sich dann ein Bild über den Durchschnittsverbrauch machen.

■ Wenn Sie sehen, daß es mit HAM nicht reicht, probieren Sie es ruhig mit 16 oder 32 Farben. Die Animationen sehen dabei oft auch nicht schlechter aus.

■ Probieren Sie es mit verschiedenen Kompressionsstufen des »Packers«. Dafür sollten Sie aber die Einzelbilder der Animation speichern (siehe Kapitel 4.1, IFF-Save-Gadget). Wem das zuviel Aufwand ist, der kann immer noch beim »Packer« Kompressionstufe 2 oder 3 einstellen. Schaden wird dies kaum und vielleicht spart es einiges an Speicher.

■ Eine letzte Möglichkeit ist das Speichern der RGB-Dateien. Falls Sie erst am Ende der Animationserzeugung feststellen, daß es mit HAM doch nicht geht, können Sie nachträglich mit »Show« alle RGB-Dateien in IFF-Dateien mit 16 oder 32 Farben umwandeln.

6 Scan

6.1 Einleitung

»Scan« ist ein neues Bildberechnungsprogramm, das als Alternative zu »Beams« gedacht ist. Es arbeitet nach dem »Scanline-Kohärenz«-Algorithmus. Dieser wurde im »Reflections-Handbuch« in Kapitel 8.3 kurz erläutert. Ein Animationssystem, das als Bildberechnungsprogramm nur einen Raytracer anbietet, wird sich vermutlich nicht allzu großer Beliebtheit erfreuen. Werden lange Rechenzeiten bei Einzelbildern durchaus noch toleriert, da sie entsprechend bessere Bilder liefern, so fallen die Rechenzeiten bei Animationen doch schwerer ins Gewicht – schließlich handelt es sich nicht mehr um ein, sondern um manchmal mehrere hundert Bilder. Bei einem schnelleren Programm nimmt man dann gerne die schlechtere Bildqualität in Kauf, zumal die Effekte des Raytracers, Schatten, Spiegelung und Lichtbrechung bei bewegten Bildern nicht mehr so deutlich in Erscheinung treten.

»Scan« schließt diese Lücke von »Reflections«. »Scan« ist in bezug auf die Eingabe kompatibel zu »Beams«. Er wird genauso aufgerufen, fragt nach den gleichen Parametern und liest die gleichen Dateien. Er läßt sich genauso vom Manager, vom CLI oder von der Workbench aus aufrufen. Texturen und Antialias werden genauso bearbeitet wie bei »Beams«.

6.2 Unterschiede zu »Beams«

Der auffälligste Unterschied ist die kürzere Rechenzeit. »Scan« rechnet drei- bis zehnmal schneller als »Beams« (je nach eingestellten »Beams«-Parametern). Die Nachteile sollten aber auch nicht verschwiegen werden.

■ »Scan« ist derzeit nicht in der Lage, Schlagschatten zu berechnen. In einer späteren Version soll diese Fähigkeit noch eingebaut werden. Dieser Mangel erschwert in einigen Situationen dem Bildbetrachter die exakte Ortsbestimmung eines Objekts – »rollt der Ball genau auf einer Fläche oder schwebt er darüber?« Zwar handelt es sich dabei nicht gerade um existentielle Fragen, aber ein einfacher Schlagschatten könnte sie beantworten.

■ »Scan« beherrscht keine Spiegelungen. Haben Sie Objekte mit spiegelnden Oberflächen, so erscheint auf ihnen nur die Hintergrundfarbe, oder im Fall einer Hintergrundtextur das Hintergrundbild.

■ »Scan« geht anders mit durchsichtigen Objekten um, als »Beams«. Während »Beams« tatsächlich den Lichtstrahl an der Oberfläche bricht, zeigt »Scan« nur, was dahinter liegt. »Scan« kann also transparente Objekte darstellen, nur werden die Objekte dahinter nicht verzerrt wie bei »Beams«.

Die Farbbilder 1 und 2 zeigen die Unterschiede von »Beams« und »Scan« anhand einer Beispielszene. Alle Bilder wurden mit Antialias berechnet. Farbbild 1 wurde mit »Beams« mit Spiegelung, Schatten und Lichtbrechung erzeugt. Man kann wohl unschwer erkennen, daß Farbbild 1 die bessere Qualität hat. Der Holztisch ist leicht spiegelnd, in der Spiegelkugel spiegelt sich der Rest der Szene. Am Schatten, den der Topf mit der Pflanze wirft, ist zu erkennen, daß der Topf über dem Tisch schwebt (mit Absicht). Die Flasche wirkt relativ plastisch.

Farbbild 2 wurde von »Scan« gerechnet. Hier fehlt der Schatten und die Oberflächen wirken stumpfer als bei Bild 1. Auch die Flasche hat sich deutlich verändert. »Scan« kann zwar keine Lichtbrechung, dafür aber Transparenz erzeugen. Man kann also durch transparente Objekte hindurchblicken. Durch die fehlende Brechung (die sich ja auch auf die Beleuchtung auswirkt), ist die Flasche hier nicht so plastisch wie in Bild 1. Daher kann man wohl »Scan« der Berechnung mit »Beams« vorziehen, wenn man keinen Schatten und keine Spiegelung braucht.

Auch die Rechenzeiten sprechen für »Scan« (alle auf einem Turbo-Amiga mit 14 MHz 68020er-Prozessor und 68881-Coprozessor): Farbbild 1 braucht mit »Beams« 120 Minuten, Farbbild 2 dagegen nur 10 Minuten mit »Scan«.

Ein ähnliches Bild wie Farbbild 2, läßt sich von »Beams« bei ausgeschaltetem Schatten und Raytrace-Tiefe 0 in etwa 30 Minuten berechnen. Allerdings ist dabei die Flasche ebenso schwarz, wie bei Bild 2 die Kugel, denn der Raytracer darf ja nicht ins Glas eindringen und muß daher die Hintergrundfarbe vom Glas verwenden. Bei einem nur mit dem 68000er ausgestatteten Amiga benötigen beide Programme etwa die drei- bis vierfache Zeit.

Der nächste Unterschied ist, daß »Scan« keiner Vorbearbeitung bedarf wie »Beams«, der ohne »Grid« nicht arbeiten kann. Allerdings erzeugt sich auch »Scan« beim Arbeiten umfangreiche Datenstrukturen. Diese werden dynamisch, also nur bei Bedarf reserviert und bei Programmende wieder freigegeben. Der Speicherbedarf ist aber nicht so hoch, wie für die Grid-Datei von »Beams«.

6.3 Aufruf von »Scan«

»Scan« wird ebenso wie »Beams« aufgerufen. Entweder über den Manager (siehe Kapitel 8) oder über CLI oder Workbench. In den letzteren beiden Fällen fragt »Scan« dann nach seinen Parametern:

- Name der Szenendatei.
- Name der Kamera- und Lichtdatei.
- Anzahl Spalten und Zeilen des Bildes.
- Schattieren (j = ja/n = nein). Dieser Parameter ist vorerst noch uninteressant, da er vom Programm noch nicht beachtet wird.
- Raytrace-Tiefe (wird ebenfalls nicht beachtet).
- Modus (wird ebenfalls ignoriert).
- Antialias-Stufe (wie bei »Beams«-Werte von 0 bis 4).
- Name der Bild-Datei.

Die Parameter, die »Scan« nicht interessieren, wurden aus Kompatibilitätsgründen beibehalten. So können Sie eine Datei, die die Eingaben für »Beams« enthält, unverändert für »Scan« benutzen. Bei einer späteren Version von »Scan« ist zudem die Einführung von Schatten geplant.

7 Play

7.1 Einleitung

»Play« ist sozusagen der Filmprojektor, der die zuvor erzeugte Animation an die Leinwand (in diesem Fall der Bildschirm) wirft. Die Arbeitsweise ist prinzipiell ziemlich einfach: »Play« lädt das erste Bild der Animationsdatei, in den Bildspeicher. Die darauf folgenden sind wegen der Deltakompression immer als Veränderung zu ihrem Vorgänger gespeichert. Das Vorgängerbild liegt also schon im Bildspeicher. Daraufhin wird eine Liste abgearbeitet, die nur besagt, welches Byte im Bildspeicher zu ändern ist, damit das Nachfolgerbild entsteht. Alle anderen bleiben unverändert. Anschließend wird mit dem nächsten Bild genauso verfahren, bis alle Bilder abgearbeitet sind.

Damit wäre die Animation normalerweise zu Ende. Der »Packer« baut aber aus jeder Animation eine Endlosschleife, die so lange wiederholt wird, bis der linke Mausknopf betätigt wird. Beim Schließen der Animationsdatei durch »Packer« wurden nach dem letzten Bild nämlich noch zusätzliche Bilddaten angefügt, die es »Play« ermöglichen, vom letzten Bild wieder zum ersten zu kommen und wieder von vorne zu beginnen.

Der Vorgang, ein Bild durch seinen Nachfolger zu ersetzen, ist nicht in beliebig kurzer Zeit möglich. Wie man sich denken kann, hängt die Dauer dieses Schrittes vom Umfang der Veränderungen pro Bild ab. Je größer die Änderung eines Bildes zu seinem Vorgänger ist, desto länger braucht »Play«, um diese Änderung im Speicher zu realisieren. Diese Tatsache hat zwei wichtige Konsequenzen:

- Ein vorgegebener Zeitrahmen für die Animation ist oft nicht einzuhalten. Wenn die Animation für 20 Bilder/Sekunde ausgelegt wurde, so kann sie nur mit dieser Geschwindigkeit abgespielt werden, wenn »Play« nicht länger als 1/20 Sekunde für einen Bildwechsel braucht. Das hängt, wie schon erwähnt, vom Umfang der Daten ab, aber auch von der Geschwindigkeit des Computers. Eine Animation, die auf einem 68020er Amiga zufriedenstellend läuft, kann auf einem normalen Amiga durchaus ruckeln, weil dieser deutlich langsamer ist.

- Ein Bild kann nicht durch seinen Nachfolger ersetzt werden, solange es am Bildschirm zu sehen ist, weil dabei der Betrachter diesen Vorgang als störend empfinden würde, und sei er auch noch so kurz. Dabei würde auf jeden Fall der Bildschirm flackern.

Um dem abzuhelfen wird bei RAM-Animationen meist die »Double-Buffering«-Technik angewandt. Man hat dabei zwei Bildspeicher gleichzeitig in Bearbeitung und läßt immer den am Bildschirm erscheinen, der gerade nicht geändert wird. Dazu müssen sich die Bilder sozusagen überkreuzen. Es entstehen zwei parallele Bildserien, die mit geraden und die mit ungeraden Bildnummern.

In Bildspeicher A sind die geraden, in Speicher B die ungeraden Bilder. Am Anfang ist A zu sehen, in dem sich Bild 0 befindet. Währenddessen wird Bild 1 nach B geladen. Sobald Bild 1 in B ist, wird B gezeigt. In dieser Zeit wird in A Bild 0 durch Bild 2 ersetzt usw. So bleibt dem Betrachter das Flackern bei den Bildänderungen weitgehend erspart.

7.2 Aufruf von »Play«

»Play« kann über den Manager aufgerufen werden (siehe dort) oder über CLI oder Workbench. Beim CLI sieht der Aufruf folgendermaßen aus:

```
»Play«   <Anim-Datei>
```

»Play« lädt dann die gesamten Animationsdaten in den Speicher und spielt sie dann ab (wenn der Speicher ausreicht). Mit den Maustasten sowie der Tastatur können Sie dann noch in gewissem Unfang Einfluß auf den Zeitablauf der Animation nehmen:

- Ein Klick mit der linken Maustaste oder Druck auf die Taste 2 veranlaßt »Play«, die Animation so schnell wie möglich abzuspielen.

- Ein weiterer Klick mit der linken Maustaste oder Druck auf die Taste 3 veranlaßt die Animation beim aktuellen Bild stehenzubleiben. Durch Klicken mit der rechten Maustaste können Sie einzelbildweise durch die Animation blättern. Ebenso durch weiteres Drücken der Taste 3.

- Der nächste Klick mit der linken Maustaste oder Druck auf die Taste 1, spielt die Animation mit normaler Geschwindigkeit weiter.

- Ein Klick der linken Maustaste, die mindestens eine Sekunde gedrückt bleibt, bricht die Animation ab. Das gleiche bewirkt die Tastenkombination »ctrl-c«.

»Play« unterstützt auch Animationen, die auf mehreren Disketten verteilt sind. Genaueres dazu finden Sie im Kapitel 8.2 unter dem Punkt »split_anim«. Der Aufruf dabei ist genau der gleiche. Bei einer geteilten Animation existiert zu <Anim-Datei> eine weitere Datei <Anim-Datei>.spl, in der »Play« dann Informationen über die weiteren Teile der Animation findet.

»Play« prüft außerdem, ob zu der Animationsdatei <name>, die er abspielen soll, auch eine Text-Datei unter dem Namen

```
<name>.txt
```

existiert. Wenn ja, zeigt er, bevor er die Animation lädt, zuerst diesen Text – man könnte dies beispielsweise für Vorspänne, Copyrightvermerke oder ähnliches verwenden.

Auch die weiteren Utility-Programme »join_anim«, »anim_time« und »merge_anim« können recht hilfreich sein. Sie sind in Kapitel 9 genauer beschrieben.

8 Der Manager

8.1 Einleitung

Der »Manager« hat hier die gleiche Funktion wie bei »Reflections«. Das Aufrufen der einzelnen Teilprogramme in der richtigen Reihenfolge, um dem Benutzer die Eingabe der wesentlichen Parameter zu ermöglichen. Die richtige Aufrufreihenfolge war dort eigentlich nur bei »Grid« und »Beams« einzuhalten. Man konnte aber auch ebensogut die einzelnen Komponenten per CLI aufrufen.

8.2 Erweiterungen zum »Reflections-Manager«

Der »Manager« ist so an den »Animator« angepaßt, daß er sowohl die Bedienung wie aus »Reflections« gewohnt beibehält, als auch die neuen Möglichkeiten der Animationserzeugung unterstützt. Das Menü des »Manager« wurde daher etwas umgestellt.

Das Projekt-Menü enthält nach wie vor nur den Menüpunkt »Progr.Ende« (alle Dateien werden in den einzelnen Programmen selbst geladen). In »Reflections« hatten die einzelnen Programme »Construct«, »Beams«, »Show« und »Get_IFF« jeweils ein eigenes Menü (mit je einem Punkt). Beim »Animator« werden Sie alle über das Menü »Programm« als Unterpunkte aufgerufen. Wie die Parameter eingegeben werden, ist im »Reflections-Handbuch« genau beschrieben. Lediglich »Scan« ist hier neu. Es wird aber genauso wie »Beams« gehandhabt. Die einzelnen Menüpunkte:

»Construct«: Unverändert aus »Reflections« übernommen. Aufruf wie gewohnt.

»Beams«: Unverändert aus »Reflections« übernommen.

»Scan«: Aus »Reflections-Animator«. Eingaben wie bei »Beams«

»Show«: Hier wird eine neue Version geliefert. Sie kann auch vom »Manager« aus eine Farbtabelle einer IFF-Datei lesen. Bei Animationen sollten aufeinanderfolgende Bilder die gleiche Farbtabelle aufweisen. Zu den gewohnten Eingabegadgets aus »Reflections« kommt hier noch eines zur Eingabe einer IFF-Datei. Die Farbtabelle dieser Datei wird automatisch als Farbtabelle für das neue Bild genommen. Die IFF-

Datei muß aber in Auflösung und mit dem neuen Bild übereinstimmen. Sonst nimmt »Show« die Farbtabelle nicht an. Wenn Sie hier nichts angeben, wird wie gewohnt für das neue Bild eine eigene Farbtabelle angelegt.

Get_IFF: Unverändert aus Reflections übernommen.

Diese Programme werden jeweils einzeln aufgerufen, wie aus »Reflections« gewohnt. Sie dienen zum Erzeugen einer Szene, zum Bildberechnen und Bildbetrachten – alles Voraussetzungen oder besser Vorarbeiten für eine Animation. Die eigentliche Animation wird aber im nächsten Menüpunkt »Anim« zusammengestellt. Dort gibt es die Menüpunkte:

Animator: Aufruf des »Animator«. Dabei muß hinter »Szene« der Name einer »Construct«-Datei angegeben werden, die als Grundlage der Animation dient. Die beiden Gadget mit Namen »Start« und »Manager« haben die gewohnte Funktion. In der Zeile darüber werden kurz Fehler erläutert, falls welche auftraten.

»PLAY_Anim«: Dieser Menüpunkt ruft den Player zum Abspielen einer Animation auf. Er benötigt hinter »Anim« den Namen der Animationsdatei, die abgespielt werden soll. Beim Anklicken von »Start« startet »Manager« das Play-Programm, das die Animation lädt, die Bilder hochzählt und dann die Animation abspielt.

9 Utilities

In diesem Kapitel werden einige Hilfsprogramme beschrieben, mit denen Sie Animationen, die mit dem »Packer« erzeugt wurden, auf die eine oder andere Art manipulieren können.

9.1 merge_anim

merge_anim dient zum Zusammenfügen zweier Animationsdateien. Der Aufruf ist denkbar einfach:

```
merge_anim  <anim_datei1>  <anim_datei2>
```

Beides müssen »Anim«-Dateien sein, die vom »Packer« erzeugt wurden. Wenn alles klappt, wird die Animation aus <anim_datei2> an die erste Datei angefügt.

Die Voraussetzung für das Abspielen einer »gemerge-ten«-Animation ist natürlich der ausreichende Speicherplatz. Die neue Animation braucht dann etwas mehr Speicher als die beiden einzelnen zusammen. Der Grund dafür ist, daß merge_anim die letzten Bilder der 1. Animation mit den ersten der 2. Animation verbinden muß. Die Verbindung erfolgt wieder durch die Delta-Kompression. Da aber die beiden Animationen normalerweise völlig verschiedene Farbtabellen haben, sind auch die Bilddaten im Speicher sehr verschieden. Daher wird die Verbindung der beiden Dateien relativ viel Speicher verbrauchen. Dies fällt aber um so weniger ins Gewicht, je größer die beiden zu verbindenden Dateien sind.

Mehr als zwei Dateien lassen sich durch mehrfache Anwendung von merge_anim verbinden. Wenn sie z.B. die Animationen a1, a2, a3 und a4 zusammenfassen wollen, gehen Sie so vor:

```
merge_anim a1 a2 merge_anim a1 a3 merge_anim a1 a4
```

Wann lohnt sich die Anwendung von merge_anim ?

1. Wenn Sie zwei Animationen mit völlig verschiedenen Körpern haben, die Sie aber nacheinander zeigen wollen. Es wäre zu umständlich, die von vorneherein im »Animator« als eine Animation zu definieren.

2. Wenn sich in einer Animation die Farben stark ändern. Dies kann durch veränderliche Lichtquellen verursacht sein, oder weil plötzlich neue Körper mit neuen Farben im Blickfeld auftauchen. Da »Packer« nur Animationen mit einer Farbtabelle erzeugt

(die Farbtabelle des 1. Bildes gilt für alle Bilder), kann dies bei starken Farbänderungen im Zuge der Animation zu schlechterer Bildqualität führen. In dem Fall ist es besser, wenn Sie die Animation in zwei oder drei Teile aufteilen. Jeder Teil hat dann seine eigene Farbtabelle, ist daher in seiner Bildqualität besser. Mit merge_anim können die Teile dann wieder zu einer Animation verbunden werden.

Natürlich müssen die beiden Animationen von gleichem Bildformat sein, als die gleiche Anzahl Zeilen, Spalten und Farbanzahl haben.

Da das erste Animationsfile aus dem Aufruf durch die Kombination beider Animationen ersetzt wird, empfiehlt es sich, beim ersten File aus dem Aufruf immer eine Kopie zu nehmen.

9.2 split_anim

Sie haben eine schöne Animation erzeugt und wollen Sie stolz Ihrem Freund auf seinem Amiga vorführen. Die Animation ist nur derart aufwendig geworden, daß sie 2 Mbyte an Speicher verbraucht. Um die Animationsdatei zu transportieren, muß sie auf Disketten verteilt werden. Diese Aufgabe übernimmt split_anim. Aufgerufen wird das Programm so:

```
split_anim  <anim_datei>  <datei_name1>  <dateiname2>  ...
```

<anim_datei> ist die originale Animationsdatei <datei_namen>. Die Namen der Dateien, auf die die Animation verteilt wird. Wichtig!! Die Dateinamen müssen den Diskettennamen enthalten (nicht den vom Laufwert), sonst findet Play oder join_anim die Dateien nicht.

Sie müssen sich ausrechnen wie viele Disketten und damit Files verbraucht werden. Auf eine gehen so etwa 850 Kbyte. Bei einer 2-Mbyte-Animation bräuchten Sie z.B. drei Disketten.

Ein Aufruf sähe z.B. so aus:

```
split_anim  dh2:grosse_animation  disk1:anim1  disk2:anim2
```

Hier ist jetzt aber einiges zu beachten:

- Halten Sie leere Disketten bereit. split_anim verteilt die Animationsdatei gleichmäßig auf die Diskettendateien. Daraus können Sie ausrechnen, wie viele Disketten Sie maximal brauchen.

- Ist die Animation kleiner als ca. 850 Kbyte, braucht sie überhaupt nicht gesplittet zu werden. Dann genügt ein einfaches File-Copy.

War split_anim erfolgreich, können Sie dann Play direkt mit

```
PLAY   <datei_name1>
```

aufrufen. <datei_name1> ist das 1. Diskettenfile, das Sie beim Aufruf angegeben haben.

Split_anim erzeugt nämlich eine kleine Informationsdatei mit dem Namen <datei_name1>.spl. Darin steht, wo Play die anderen Dateien der Animation findet. Existiert zu einer Datei eine entsprechende .spl-Datei, weiß Play, daß es sich um eine gesplittete Datei handelt und verfährt entsprechend.

9.3 join_anim

join_anim hat eine ähnliche Aufgabenstellung wie merge_anim, nämlich das Zusammenfügen von Animationsdateien, jedoch mit unterschiedlichen Ausgangssituationen. Merge_anim verbindet unabhängige, vollständige Animationsdateien.

Join_anim hingegen arbeitet nur mit den gesplitteten Dateien, die von split_anim erzeugt wurden. Wozu aber join_anim, wenn man Play auch mit einer gesplitteten Datei aufrufen kann?

Das Lesen der Daten von Diskette zeichnet sich nicht gerade durch übermäßige Geschwindigkeit aus. Wenn Sie also eine Festplatte besitzen (bzw. Ihr Freund, dem Sie die Animation auf Disketten gebracht haben), ist es sinnvoll, die Animation wieder dorthin zu speichern. Andernfalls warten Sie bei jedem Start von Play einige Zeit und müssen sich zudem noch als Diskjockey betätigen.

Der Aufruf:

```
join_anim   <split_datei_1>   <anim_datei>
```

<split_datei_1> ist die erste der gesplitteten Dateien (mit der Sie auch Play aufrufen würden). Dazu muß die Datei <split_datei_1>.spl existieren.

Die, wieder zusammen gefaßte Animation wird nach <anim_datei> gespeichert.

9.4 anim_time

anim_time erlaubt die Veränderung der Abspielgeschwindigkeit.

Aufruf:

```
anim_time   <anim_datei>
```

<anim_datei> ist der Name der Animationsdatei (keine gesplittete).

Bei Aufruf lädt anim_time die Daten der Animation, die den zeitlichen Ablauf betreffen. Dann öffnet sich ein Fenster. In ihm ist ein großes Feld, in dem die Zeitdauer der einzelnen Bilder als Punkte dargestellt ist. Darunter sind zwei Gadgets:

CANCEL: Zum Abbruch des ganzen Vorgangs.

SAVE: Zum Abspeichern gemachter Änderungen.

Mit der Maus können Sie dann die Zeitpunkte der einzelnen Bilder ändern: Fahren Sie einfach zu den betreffenden Punkten, drücken Sie die linke Maustaste und verändern Sie durch Hoch- und Runterfahren mit der Maus die Zeitdauer des Bildes.

Unten im Fenster wird dann gleichzeitig die gerade bearbeitete Bildnummer, sowie die eingestellte Verzögerungszeit (= Delay) angezeigt und zwar in Einheiten zu 1/60 Sekunde.

Vom »Animator« her, haben die Bilder ja alle die gleiche Dauer, die sich nach der Animationsdauer und -Bildanzahl richtet. Hier können Sie nun das Zeitverhalten bestimmter Animationspassagen nachträglich verändern. So ist es oft sinnvoll, den Schluß der Animation zu verlangsamen, oder auch den Anfang. Oder bestimmte Bilder der Animation sollen aus dramaturgischen Gründen einige Augenblicke angehalten werden.

Sind Sie mit den Änderungen zufrieden, klicken Sie *Save* an. Daraufhin wird die Animationsdatei korrigiert. Diese kann nun wieder abgespielt, betrachtet und gegebenenfalls weiter verändert werden.

10 Hintergrundinformationen

Dieses Kapitel ist für das Verständnis der Funktionsweise des »Reflections-Animator« eher nebensächlich. Es enthält einige Hintergrundinformationen zur Computer-Animation.

10.1 Wurzeln

Computeranimation ist ein Bereich der Computergrafik, bei dem Techniken und Verfahren aus vielen anderen Gebieten zusammenfließen, sowohl aus dem technischen Bereich, aber ebenso aus dem künstlerisch-gestaltendem Bereich.

10.1.1 Bilderzeugung

Bilderzeugung ist die Voraussetzung für Animation. Die Grundlagen der meisten Verfahren zur Bilderzeugung wurden in den 60er Jahren erarbeitet. Anfangs lag der Schwerpunkt auf der korrekten geometrischen Darstellung dreidimensionaler Szenen. Hauptproblem dabei war das Erkennen und Eliminieren verdeckter Linien und Flächen. Später, mit zunehmender Rechenleistung, stiegen auch die Anforderungen über die reine Geometrie hinaus zu realistischer Simulation des Lichtverhaltens. Zunehmend wurde »Fotorealistik« angestrebt.

10.1.2 Simulation

Die ersten Interessenten an Computergrafik waren Luft- und Raumfahrt. In diesen Bereichen wurden zu Ausbildungs- und Übungszwecken Simulatoren gebraucht. In solchen Simulatoren können Menschen gefährliche Situationen trainieren, die in der Realität nicht so risikolos nachvollziehbar wären. Der Mensch nimmt einen Großteil seiner Umweltinformationen über das Auge auf, daher das große Interesse an Computergrafik. Die Anforderungen bei Simulatoren sind hohe Geschwindigkeit beim Bildaufbau und möglichst große Realität der Bilder. Bilder sollten mindestens mit einer Geschwindigkeit von 20 bis 30 Bildern pro Sekunde gezeigt werden. Daher ist die Bildqualität bei Simulatoren weit unterhalb der mit Computern erzeugbaren möglichen Qualität – denn die bestmögliche Qualität kann einfach nicht schnell genug angezeigt werden. Das ist aber oft nicht so störend wie man vielleicht annimmt. Allein die Bewegung erhöht schon den realistischen Eindruck eines Bildes.

Ein noch größerer Unterschied zur Computer-Animation ist das zeitliche Verhalten von Objekten. Während bei der Animation diese hauptsächlich vom »Animator«, also

einem Menschen interaktiv am Rechner bewegt werden, ist das Verhalten von Objekten des Simulators normalerweise programmiert.

10.1.3 Spiele

Spiele kann man als Minimal-Varianten der Simulation bezeichnen. Allerdings sind hier Spiele gemeint, bei denen sich etwas am Bildschirm bewegt, also Action-Spiele, durchaus auch Baller-Spiele, Flugsimulatoren etc. Sie haben meist ähnliche Anforderungen nach schnellem Bildaufbau wie Simulatoren – wenn auch nicht ganz so strenge. Deswegen können auch einfachere Algorithmen zum Bilderzeugen verwendet werden.

Auch hier hat sich im Laufe der Jahre sehr viel getan. Man denke an den Anfang der Computerspiele, das legendäre »Pong«. Es zeugt von der damaligen rein Text-orientierten Computerszene, daß ein grafisch so simples Spiel, bei dem sich nur 3 Punkte über einem schwarzen Schirm bewegen, solchen Erfolg verzeichnen konnte.

»Pong« war der erste Vertreter der 2-D-Spiele. Anschließend folgte eine lange Reihe von 2-D-Spielen, die mit der Zeit so ziemlich alle Aspekte von zweidimensionalen Aktionen enthielten, die man sich denken kann. Obwohl nach wie vor neue 2-D-Spiele mit immer perfekterer Grafik erzeugt werden, geht der Trend doch zunehmend zu den 3-D-Spielen über. Der zeitlich spätere Beginn von 3-D-Spielen leuchtet ein, da die Bildberechnung hier viel aufwendiger ist, als bei 2-D-Spielen. Daher mußten sich 3-D-Spiele lange Zeit mit sehr einfachem Bildaufbau begnügen (beispielsweise nur Strichgrafik wie bei vielen Flugsimulatoren). Erst in letzter Zeit kommen mit zunehmender Rechenleistung der Heimcomputer 3-D-Spiele auf den Markt, die es in der Qualität durchaus mit teuren Flugsimulatoren früherer Jahre aufnehmen.

10.1.4 Zeichentrickfilm

Sehr viele Konzepte aus dem Zeichentrickfilm werden ebenso in der Computer-Animation übernommen. Dies ist nicht verwunderlich, hat doch der Zeichentrick sehr ähnliche Grundvoraussetzungen. Der Zeichner hat hier völlige Freiheit über Bewegungen, Verformungen, generell alle Aktionen, die er an Objekten vornehmen möchte. Das Erstellen von Zeichentrickfilmen ist aber ungleich mühsamer als bei Computeranimationen, da alle Einzelbilder von Hand gezeichnet werden müssen. Um hier Zeit und vor allem Geld zu sparen, wird die Keyframe-Technik angewendet. Die erfahrenen Zeichner und Animateure zeichnen nur noch die Figuren für ganz bestimmte Schlüsselbilder (»Keyframes«). Die Zwischenbilder können dann von weniger erfahrenen (und damit auch weniger teuren) Zeichnern erstellt werden. Diese Keyframe-Technik wird heute auch von Computer-Animationssystemen angewendet. Der Regisseur definiert nur die Position von Objekten zu bestimmten Zeitpunkten. Die Rolle der Hilfszeichner wird vom Computer übernommen.

10.2 Interaktive und programmierbare Animations-Systeme

An die Bewegung dreidimensionaler Objekte kann man von verschiedenen Seiten herangehen, je nachdem welche Anforderungen zugrundeliegen. Das interaktive Animieren wird von sogenannten Keyframe-Systemen unterstützt. Der »Animator« bewegt interaktiv am Bildschirm Objekte und baut daraus Bewegungspfade zusammen. Ein anderer Weg wäre das explizite Programmieren einer Bewegung. Beide Wege haben ihre Vor- und Nachteile.

10.2.1 Keyframe-Systeme

Der »Animator« gibt nur Position und Orientierung von Objekten zu bestimmten Zeitpunkten an. Dies geschieht in der Regel einfach durch Bewegen der Objekte, beispielsweise mit der Maus, ist also relativ einfach. Der Regisseur sieht am Bildschirm sofort die richtige Position. Danach wird diese Position entweder einem Bild, oder einem Zeitpunkt zugeordnet. Der Rechner kann dann die Positionen in den Bildern dazwischen (»in-betweens«) interpolieren. Dies bedingt ein stark interaktives Arbeiten mit dem Animations-System. Der unerfahrene Benutzer wird unterstützt, da er sich durch ständiges Ausprobieren und Verbessern dem gewünschten Resultat annähern kann.

Solche »Animatoren« werden in der Regel mit Menüs, Icons und Gadgets gesteuert. Das ermöglicht das einfache Arbeiten, hat aber auch einen Nachteil: Das Animations-Programm stellt eine Anzahl von Operationen zur Verfügung, evtl. mit verschiedenen Optionen. Diese sind alle einzeln beispielsweise über Menüs aufrufbar. Wenn es aber darum geht, verschiedene Operationen zu kombinieren oder zu neuen Operationen zusammenzufassen, sind Menü-gesteuerte Programme oft überfordert. Es geht natürlich bei geeigneter Programmierung schon, nur verliert dann das Programm seine Übersichtlichkeit. Hinzu kommt, daß der Mensch Schwierigkeiten bekommt, wenn er mehr als sechs bis acht Objekte oder Bewegungen gleichzeitig am Bildschirm kontrollieren soll. Er verliert einfach irgendwann ganz einfach die Übersicht.

Keyframe-Systeme sind daher eher für nicht allzu komplexe Animationen geeignet. In diesem Bereich sind sie allerdings sehr effektiv.

10.2.2 Programmierbare Systeme

Im Gegensatz zu Keyframe-Systemen wird hierbei die Bewegung oder Veränderung von Objekten explizit programmiert. Dazu gibt es manchmal eigene Grafik-Programmiersprachen. Meistens werden jedoch verbreitete Programmiersprachen benutzt, wie etwa C, Fortran oder Pascal, die über Zusatzfunktionen Zugriff zu grafischen Objekten haben.

Ein Hauptnachteil sticht sofort ins Auge. Für eine bestimmte Bewegung muß extra ein Programm geschrieben und kompiliert werden. Erst nach Ausführung des Programms kann die programmierte Bewegung begutachtet werden. Meistens ist man nicht zufrieden, daher muß das Programm geändert und neu kompiliert werden. Jeder, der schon einmal programmiert hat, kennt diesen lästigen Vorgang. Bei Bewegungen, die mit einem Keyframe-System auch ausführbar sind, ist die Programmierung nicht angebracht, weil dabei das Erzeugen, Begutachten und Modifizieren von Bewegungen einfach zu umständlich ist. Anders sieht es bei komplexen Bewegungen aus. Die sind zwar aufwendiger zu programmieren, aber mit Keyframe-Systemen manchmal einfach nicht machbar.

Programmiersprachen haben flexible Kontrollstrukuren eingebaut, die genau das erlauben, was Keyframe-Systeme normalerweise vermissen lassen, nämlich das beliebige Zusammenfassen von Funktionen, das beliebige Wiederholen von Funktionen für verschiedene Objekte. Daneben lassen sich dann auch eigene Operationen programmieren, die das Animations-System nicht anbietet.

Voraussetzung ist dabei natürlich Programmiererfahrung und ein sehr gutes räumliches Vorstellungsvermögen. Komplexe Animationen bedingen komplexe Programme, und die müssen erst geschrieben werden. Der Programmierer muß sich die Bewegungen, die er gerade programmiert, sehr genau vorstellen können, denn ausprobieren kostet aus naheliegenden Gründen immense Zeit.

Im natur- oder ingenieurwissenschaftlichen Bereich werden in der Regel programmierbare Systeme vorgezogen. Erstens haben die Anwender häufig Programmiererfahrung. Zweitens sind die benötigten Bewegungen oft durch mathematische Formeln beschrieben, was automatisch die Programmierung nahelegt.

10.2.3 Script-Systeme

Einige Vorteile von beiden zuerstgenannten Systemen bieten Script-orientierte Systeme. Sie haben eigene Script-Sprachen, die man im Prinzip auch als einfache Programmiersprachen bezeichnen könnte. Eine Script-Sprache bietet beispielsweise für jede Operation oder Funktion eines Animations-Systems ein Text-Kommando an. Beispiele wären etwa Kommandos wie »Bewege x,y,z«, »Drehewinkel,achse« oder »Vergrößere Faktor«. Diese Kommandos können dann in Texteditoren eingegeben werden, wie ein ganz normales Programm. Das Animations-System kann solche Textdateien mit Kommandos einlesen und ausführen.

Es ist dann auch möglich, die Kommandos selbst über Tasten im Animations-System einzugeben, wo sie sofort ausgeführt werden. Der Anwender hat dann ständig die Übersicht, was ein bestimmtes Kommando bewirkt.

Der Nachteil gegenüber einer »richtigen« Programmiersprache ist die geringere Flexibilität. Script-Sprachen haben in der Regel nicht so viele Kontrollstrukturen und

Möglichkeiten wie herkömmliche Programmiersprachen, bieten aber schon mehr Möglichkeiten als die reinen Keyframe-Systeme.

Der Idealfall sind natürlich Kombinationen der drei Grundtypen. Keyframe-Systeme haben eine eingebaute Script-Sprache. Die Verbindung von Keyframe-Systemen mit Programmiersprachen ist dagegen problematischer.

Viele Operationen, die mit Keyframe-Systemen ausgeführt werden, sind stark interaktiv geprägt, d.h., sie liefern dem Benutzer während der Operation viele Zusatzinformationen. Das System meldet, wenn eine Operation nicht möglich ist und verhindert soweit wie möglich Bedienungsfehler. Will man die gleichen Operationen per Programm ansprechen, so fehlt dieser Feedback.

10.3 Simulation und Robotik

Oft haben Animationen nicht den Anspruch, bis ins letzte Detail realistisch zu sein. Das betrifft nicht unbedingt den Bildrealismus, sondern eher das Bewegungsverhalten von Objekten. Bei der Bildqualität geht das Bestreben natürlich zu immer realistischerem Aussehen. Die Bewegungen sind hingegen oft nicht realistisch, was beabsichtigt sein mag, um gewisse Dinge besonders zu betonen. Es sieht beispielsweise oft besser aus, wenn Bewegungen oder Abläufe übertrieben werden (Verformen, Verzerren von Körpern bei Beschleunigung). Es ist zwar nicht sehr realistisch, aber es leitet das Auge des Betrachters, um diese Dinge in der Animation zu betonen oder unterstreichen.

Bei Simulationen ist hingegen möglichst hoher Realismus bei den Bewegungen gefragt. Simulationen werden meist in der Industrie zu Forschungs- und Demonstrationszwecken angewendet. Beispiele sind etwa:

- Die Simulation des Verhaltens von Molekülen unter verschiedenen Bedingungen
- Verdeutlichung der Funktionsweise eines Industrieroboters.
- Simulierte Crashtests in der Automobilindustrie.

Solche Simulationen werden in der Regel mit programmierbaren Systemen erzeugt, da das simulierte Verhalten sehr speziellen Gesetzen folgt, die ein allgemeines Animations-System zwangsläufig überfordern würden. Hier liegt es nahe, solche Animationen selbst zu programmieren.

An dem Beispiel des simulierten Industrieroboters läßt sich schon ein weiterer verwandter Bereich erkennen, die Robotik. Dieses ist ein Bereich, bei dem sich Informatik und Maschinenbau überschneiden. Das Programmieren von Robotern ist ein Gebiet, in dem intensiv geforscht wird. Viele Bereiche der Informatik und anderer Ingenieurwissenschaften fließen hier ein, wie etwa Mustererkennung, künstliche Intelligenz, Meß- und Regelungstechnik.

Gewisse Anforderungen, die vom Anwender an Roboter gestellt werden, lassen sich auch bei der Animation erkennen. Dies betrifft vor allem die Bewegung von Robotern. Der Anwender teilt dem Roboter beispielsweise mit: Bewege den Greifer von Punkt A nach B. Hängt der Greifer an Roboterarmen, die über Gelenke verbunden sind, muß die Greiferbewegung in entsprechende Gelenk-Drehungen umgerechnet werden.

Gerade diese Problemstellung wird oft auch von Animations-Systemen aufgegriffen. Der Rechenaufwand nimmt aber mit der Anzahl der Gelenke sehr stark zu, so daß das Verfahren dann unpraktikabel wird. Andere angrenzende Themen aus der Robotik wären Bewegungsplanung, Kollisionserkennung und die Berücksichtigung von Dynamik, also Gewicht, Geschwindigkeit, Trägheit von Körpern.

Gerade bei der Entwicklung von Roboterprogrammen wird oft zum Mittel der grafischen Simulation gegriffen. Roboterprogramme können so schon getestet und beurteilt werden, bevor der dazugehörige Roboter überhaupt gebaut ist. Die Programme sprechen anstelle der Roboterhardware ein angepaßtes Animations-System an. Das erzeugt die Simulation dann entweder in Echtzeit, wenn es dazu in der Lage ist, oder mit Verzögerung als Animation, die man sich später anschauen kann.

10.4 Weitere Konzepte

10.4.1 Zielorientierte Animation

Mit zunehmenden Fähigkeiten der Grafiksysteme steigen auch die Ansprüche. Man möchte nicht nur einfache Animationen, bei denen ein paar Teile irgendwie herumwirbeln, sondern komplexe Szenerien, die realistisch aussehen und sich auch realistisch bewegen. Je realistischer oder natürlicher eine Bewegung wirken soll, desto mehr Arbeit und auch Erfahrung wird in der Entwurfsphase benötigt.

Forschungen gehen daher in die Richtung, den »Animator« davon zu befreien, sich bei jeder kleinen Bewegung, um die Natürlichkeit kümmern zu müssen. Angestrebt ist dabei eine zielorientierte Arbeitsweise etwa in der Art: Man gibt einem Objekt beziehungsweise Körper das Ziel, zu einem bestimmten Zeitpunkt am Ort X zu sein. Wie es dahin kommt, ist Sache des Animations-Systems.

Handelt es sich beispielsweise um einen Roboter, so sollte das Animations-System automatisch dafür sorgen, daß er in Schritten über einen Boden läuft, Auswirkungen der Schwerkraft berücksichtigen (vielleicht ein gebeugter Gang), über Hindernisse hinwegsteigt etc.

Eine solche Art der Animationserzeugung ist wohl der Traum jedes Trickfilm-Regisseurs, aber da ist noch einiges an Arbeit zu investieren. Es läuft praktisch auf die Simulation einer ganzen Welt, einschließlich Naturgesetzen (reale oder künstlich entworfene) hinaus.

Das ist heutzutage durchaus schon möglich, wenn man die Komplexität der simulierten Welt einschränkt. Simulatoren und auch viele Computerspiele haben diese Konzepte schon ansatzweise integriert. In einem Flugsimulationsspiel wird beispielsweise der Programmteil, der die gegnerischen Flugzeuge leitet, nur Kommandos an die Flugzeuge geben (fliege zu Punkt x etc) und sich nicht um die Details der Ausführung kümmern. Ein anderer Programmteil übernimmt dann die Feinarbeit, sorgt dafür, daß sich das Flugzeug neigt, den Kurs ändert, steigt, sinkt (auch dieses können Kommandos sein, die wiederum von noch untergeordneteren Instanzen detaillierter ausgeführt werden).

Man ist heutzutage bei solchen zielorientierten Systemen noch auf einfache »Welten« mit einfachen »Naturgesetzen« beschränkt, was sich aber mit der zunehmenden Leistungsfähigkeit der Computer sicherlich ändern wird.

10.4.2 Partikel-Systeme

Ein anderes Konzept greift das Problem »flauschiger« Objekte auf. Man bekommt mit einem normalen Animations- oder Grafiksystem schnell Probleme, wenn man Objekte erzeugen will, die keine exakten Grenzen haben, sondern diffus im Raum verteilt oder haarig sind, sich jedenfalls schlecht durch Polygone oder andere Grundobjekte beschreiben lassen.

Um solche Objekte zumindest annähernd zu simulieren und darzustellen, wurde das Konzept der »Partikel-Systeme« entwickelt. Flauschige Objekte werden dabei durch eine große Anzahl punktförmiger Partikel simuliert. Ein Beispiel wäre Nebel. Nebel besteht ja aus unzähligen kleinsten, schwebenden Wassertropfen. Diese Tröpfchen könnte man durch die Partikel simulieren.

Da jedes Partikel nur sehr wenig Daten braucht (im Prinzip nur Koordinaten und eventuell einige Attribute), kann man sehr viele von ihnen gleichzeitig handhaben. Auf jeden Fall viel mehr als Dreiecke, Kugeln oder Zylinder. Alleine die Partikel reichen aber nicht aus. Um aus einer Partikelmenge etwas Konkretes zu erzeugen, eine Wolke, Nebel oder Feuer, müssen die Partikel auch angeordnet und/oder bewegt werden. Bei Zigtausenden von Partikeln ist der »Animator« sicher überfordert. Die Steuerung der Partikel geschieht daher über Programme.

Die Partikel werden dabei je nach simuliertem Phänomen unterschiedlich gesteuert. Man kann Teilchen abhängig von ihrer Position, Geschwindigkeit manipulieren. Man kann Eigenschaften benachbarter Teilchen berücksichtigen. Teilchen können »sterben« oder neu »geboren« werden. Oft wird auch ein Zufallsfaktor eingebaut.

Für solche Partikelprogramme muß man sich sehr eingehend mit dem, zu simulierenden Phänomen beschäftigen. Unterschiedliche Phänomene dürften auch radikal verschiedene Programmlösungen erfordern. Man kann als »Animator« auch nicht

genau bestimmen oder vorhersehen, wie sich ein Partikelsystem verhält. Das Erzeugen einer Wolke, die ganz genau ein bestimmtes Aussehen haben soll, ist mit Partikel-Systemen kaum machbar, vielmehr gibt man in dem Beispiel globale Eigenschaften der Wolke an, wie Rauhigkeit, Durchsichtigkeit, Farbe etc. und hofft, daß die Wolke dann ein akzeptables Aussehen bekommt. Anwendungen dieser Art sind daher meist auf programmierbare Animations-Systeme beschränkt (wenn sie überhaupt heutzutage schon angeboten werden).

10.5 Körperanimation

Die Simulation und Animation von Funktionen und Eindrücken des menschlichen Körpers, gehört zu den schwierigsten Aufgabengebieten der Computer-Animation. Die Komplexität dieser Aufgabe ist eng verbunden mit der Komplexität des menschlichen Körpers, mit der Flexibilität (ein menschlicher Körper ist nicht aus starren Teilen aufgebaut), aber auch mit dem kritischen Auge des Betrachters, denn hier ist vielmehr Realismus gefragt, als bei technischen Objekten, bei der schon eher eine vereinfachte Darstellung akzeptiert wird.

Bei einer Animation mit menschlichen Körpern fallen Unregelmäßigkeiten viel stärker auf, als wenn irgendwelche technischen Objekte animiert werden. Das liegt auch daran, daß der Betrachter selbst ein Mensch ist und ganz genau weiß, wie er sich seine Bewegungen vorstellt. Bei einem gehenden Roboter ist man eher bereit eine weniger realistische Schrittfolge zu akzeptieren; es könnte ja sein, daß dies auf die speziellen Eigenschaften von Robotern zurückzuführen ist. Bei einem Menschen sieht man dagegen sofort, wenn etwas nicht stimmt.

Der »Animator« muß daher sehr viel Sorgfalt in die Bewegungsdefinition legen. Hier kommt die Komplexität des menschlichen Körpers erschwerend hinzu. Um die Bewegungen einigermaßen realistisch aufzubauen, muß der Körper aus etwa 200 Teilkörpern zusammengesetzt sein, die mit Gelenken verbunden sind.

Diese Körperanzahl ist eigentlich schon jenseits der Grenze dessen, was mit einem interaktiven Keyframe-System gehandhabt werden kann. Das Animations-System muß jedenfalls sehr leistungsfähig und flexibel sein, um bei dieser Aufgabe nicht zu versagen.

Animationen von menschlichen Körpern werden häufig auch mit programmierbaren Animations-Systemen (siehe Kapitel 10.2.2) erzeugt. Dabei wird der Bewegungsablauf von Programmen erzeugt, die den menschlichen Körperbau simulieren und beschränkte Kenntnisse der Umgebungswelt besitzen. Es gibt auch schon einfache Zielorientierte Systeme, wie in Kapitel 9.4 beschrieben. Meist wirken solche synthetischen Bewegungen aber noch zu steril, und damit unecht. Man steckt hier eben noch in den Kinderschuhen.

Eine andere Methode erzielt dagegen recht realistische Ergebnisse. Dabei werden einfach Bewegungen von echten Menschen per Kamera aufgezeichnet und dann auf die, im Computer modellierten Menschen umgerechnet. Dies kann auch schon recht komplex werden, je nach Art der Bewegung. Unter Umständen muß der Mensch mit drei Kameras gleichzeitig aufgenommen werden, um der dritten Dimension gerecht zu werden. In verschiedenen Universitäten (unter anderem Mainz, Tübingen und München) wird diese Technik bereits in der Biomechanik eingesetzt. Wenn man noch weiter geht, kann man auf die Kameraaufzeichnung verzichten und die Bewegung direkt übertragen:

Ein menschlicher Darsteller hat dabei einen »Datenanzug« an, der mit Meßfühlern belegt ist. Diese Meßfühler registrieren an möglichst vielen Punkten des Körpers die Bewegungen, die Drehungen der Gelenke und übertragen Sie an einen Rechner, der sie beispielsweise sofort an einem Computer-Modell nachvollziehen kann. Auf diese Weise sind auch sehr komplexe Bewegungsabläufe wie Sprünge, Tänze etc. machbar.

10.6 Hardware-Aspekte

Dieses Thema läßt sich einerseits recht kurz abhandeln: Je schneller der Rechner, desto besser. Andererseits ist damit kaum jemandem gedient. Denn dieses Motto könnte man auf jeden Computer-Bereich anwenden, für Grafik und Animation gilt es sogar mit drei dicken Ausrufezeichen. Das Erzeugen realistischer Bilder und das Animieren solcher Bilder ist extrem rechenintensiv. Aber wie kommt das?

Man kann grob zwischen den Bereichen Bilderzeugen und Modellieren unterscheiden. Für beide gelten unterschiedliche Voraussetzungen, obwohl diese oft verschwimmen, wenn auf demselben Rechner modelliert und Bilder berechnet werden.

Modellieren: Das Entwerfen von Objekten und Animationen ist eine stark interaktive Aufgabe. Der Designer braucht schnellen Feedback von seinen Aktionen. Die Interaktion geschieht überwiegend grafisch. Dabei ist vor allem der Rechner für eine schnelle grafische Ausgabe gefordert. Gerade diese Aufgabe erzwingt im professionellen Bereich spezialisierte Grafikrechner. Sie sind unter der Bezeichnung »Grafik-Workstations« bekannt. Ihre Stärken liegen zum einen bei der Bilddarstellung (hohe Bildschirmauflösungen von mindestens 1000 mal 1000 Pixel sind Standard, dazu kommen eine Farbtiefe von 24 Bit, also 16 Millionen Farben und ein schneller Bildaufbau durch spezialisierte Chips). Spezialisierte Chips hat auch der Amiga. Bei den Grafik-Workstations sind inzwischen Chips verfügbar, die nicht nur Linien zeichnen oder Flächen füllen können, sondern sogar komplexe 3-D-Objekte mit mehreren tausend Polygonen in Echtzeit schattiert und beleuchtet darstellen können.

Zum anderen sind Grafik-Workstations auch auf die Manipulation von 3-D-Objekten spezialisiert. Für diese Aufgabe haben sie Spezialchips, die beispielsweise das Rotie-

ren, Verschieben, Vergrößern von 3-D-Objekten sehr viel schneller vornehmen können als die normale CPU. Mit dieser Ausstattung sind solche Workstations in der Lage Wire-Frame- oder sogar schattierte Animationen, auch von komplexen Objekten, in Echtzeit zu berechnen und darzustellen. Eine enorme Erleichterung für den »Animator«, bekommt er doch einen sofortigen Überblick über erzeugte Objekte und Bewegungen.

Die Rechenleistungen, die Workstations zur Echtzeitanimation liefern, nützen aber in der Regel wenig, wenn man Raytracing-Qualität erreichen will. Die Grafik-Chips sind dafür zu spezialisiert. Für Raytracing oder ähnlich qualitative Algorithmen muß normalerweise die CPU benutzt werden. Das wird auch gemacht, denn auch die Prozessoren der Workstations sind nicht gerade langsam. Wenn aber harte Zeitanforderungen herrschen, wird die Bilderzeugung gelegentlich auf noch stärkere »Number-Cruncher« verlagert.

Bilderzeugen: Die Bildberechnung ist von der Rechenzeit her der aufwendigste Teil einer Animation. Bei ihr werden eine Unzahl mathematischer Operationen ausgeführt. Für ein typisches Raytrace-Bild, das in LoRes-Auflösung mit Beams in drei Stunden gerechnet wurde, werden etwa 50 bis 100 Millionen Fließkomma-Operationen ausgeführt.

Hier ist natürlich ein möglichst schneller Rechner gefragt. Einschränkend kommt im allgemeinen aber der Kostenfaktor hinzu. Die schnellsten Rechner sind in der Regel die teuersten. Animationsstudios, die an ihrer Arbeit Geld verdienen wollen, müssen hier oft sehr genau überlegen und kalkulieren, welcher Rechner für ihre Belange der idealste ist. Ein sehr schneller Rechner kann unter Umständen ungeeignet sein, weil seine laufenden Kosten die Animationen so verteuern, daß sich der Geschwindigkeitsgewinn nicht auszahlt. Ein zu langsamer ist dagegen trotz seiner geringen Kosten auch nicht geeignet, weil Termine nicht eingehalten werden können. Einige Überlegungen zu den heute verwendeten Rechnertypen könnten ganz interessant sein.

Es wird ständig nach Erhöhung der Rechengeschwindigkeit gestrebt, nicht nur von seiten der Computer-Grafik, da gibt es Bereiche, die sind noch viel »heißer« auf Rechenpower. Höhere Rechengeschwindigkeit wird auf zwei Arten angestrebt:

- Konventionelle Rechner mit höherer Arbeitsgeschwindigkeit.
- Parallelrechner.

Man ist sich einig, daß der zweite Ansatz langfristig der vielversprechendere ist. Erstens läßt sich die Arbeitsgeschwindigkeit nicht immer weiter erhöhen, ohne daß die Chips durchbrennen. Technologiebedingt stößt man irgendwann auf Grenzen, die nicht mehr zu überschreiten sind. Elektrische Signale lassen sich nicht schneller als mit Lichtgeschwindigkeit übertragen. Ebenso lassen sich die Schaltzeiten von Transistoren nicht beliebig verkürzen.

Zweitens kommt die Erhöhung der Arbeitsgeschwindigkeit ja genauso Parallelrechnern zugute. Der Nachteil bei den Parallelrechnern ist die Neustrukturierung vorhandener Software. Damit ist nicht nur eine gewisse Umprogrammierung gemeint, sondern ein radikales Umdenken bei der Formulierung von Algorithmen.

Meistens läßt sich ein Parallelrechner nur effektiv verwenden, wenn die Algorithmen speziell parallelisiert wurden. Es kommt vor, daß konventionelle Software auf einem Parallelrechner langsamer läuft, als auf einem konventionellen Rechner, weil die Parallelfähigkeiten nicht ausgenutzt wurden und der für die parallelen Prozessoren nötige Verwaltungsaufwand mit zu Buche schlägt.

Auch die Berge von Software, die sich inzwischen aufgetürmt haben, müßten größtenteils neu geschrieben werden, was angesichts der Kosten sicher noch einige Zeit auf sich warten lassen wird.

Zudem gibt es sehr viele verschiedene Ansätze zur Parallelität. Da wären einmal Rechner mit wenigen sehr starken Prozessoren, dann Rechnern mit Spezialprozessoren, die eine Anweisung auf einer Vielzahl von Daten gleichzeitig ausführt, schließlich Rechner mit sehr vielen, aber leistungsschwachen Prozessoren, Netzwerke, Transputer u.v.m. Für jeden dieser Bereiche gibt es Anwendungen, die schon erfolgreich parallelisiert wurden. Der ideale, möglichst allgemein gültige Weg ist aber noch nicht in Sicht.

Man kann aber auch jetzt schon Mittelwege beschreiten, d.h., schon mit konventionellen Programmen parallel rechnen. Man nimmt einfach mehrere Rechner und läßt das gleiche Programm auf allen laufen. Computer-Animation eignet sich hierfür besonders. Die Einzelbilder werden nach wie vor konventionell, also sequentiell berechnet. Die Animation als Ganzes, entsteht aber parallel, da immer so viele Bilder gleichzeitig entstehen, wie Rechner zur Verfügung stehen.

Ein Beispiel für die Parallelisierung einer Computer-Animation ist das Projekt »Occursus cum novo« der Universität Karlsruhe. Dabei handelt es sich um eine ca. fünfminütige Raytrace-Computer-Animation, die auf einem Unix-Netzwerk von 30 SUN-Workstations erzeugt wurde. Ziel des Projekts war die Erzeugung einer Raytrace-Animation von nichttrivialer Länge in vernünftiger Zeit mit vernünftigen Kosten. Erzeugt wurde die Animation mit dem Raytrace-Programm »VERA«, das am Institut für Betriebs- und Dialogsysteme, Abteilung Computergrafik entwickelt wurde. Die Animation besteht aus 7550 Einzelbildern, von denen etwa 4000 tatsächlich berechnet wurden, die anderen waren Wiederholungen. Die Rechenzeitsumme für die Erzeugung aller Bilder belief sich auf den 68020-basierten SUN-Rechnern auf fast drei Jahre. In dem Netzwerk von 30 Rechnern wurde die Animation in drei Monaten fertiggestellt.

Dabei mußten einige Randbedingungen beachtet werden:

- Das Rechnernetz ist über mehrere Institute verteilt. Die normalen Benutzer des Netzes sollten so wenig wie möglich gestört werden.
- Raytrace-Programme verbrauchen normalerweise viel Speicher und Rechenzeit. Sie sollten daher nicht laufen, wenn die Workstations anderweitig benutzt waren.
- Die ganze Animation sollte weitgehend automatisiert werden.

In jedem der Netzwerk-Rechner läuft ständig ein Hintergrundprozeß, der, sobald der Rechner frei ist, d.h. kein Benutzer damit arbeitet, einen neuen Bild-Auftrag anfordert und die Bildberechnung startet. Er bekommt die nötigen Daten von einem zentralen Rechner, der den Ablauf der gesamten Animationsberechnung überwacht. Sobald ein Benutzer wieder an einem der Netzwerk-Rechner arbeiten will, wird die Bilderzeugung (ähnlich wie bei »Beams«) abgebrochen, so lange, bis der Rechner wieder frei ist. Ist ein Bild fertig, wird es an den zentralen Rechner übertragen, wo es dann auf Magnetband gesichert wird.

Dazu kommen dann noch viele Verwaltungsprobleme. So mußte Protokoll über fehlerhafte oder abgebrochene Bildberechnungen geführt und auf außergewöhnliche Störungen wie Stromausfall oder Abschalten des Rechners durch den Benutzer, angemessen reagiert werden.

Das Projekt hat verdeutlicht, daß mit Netzwerken auch sehr aufwendige Raytrace-Animationen in akzeptabler Zeit zu erzeugen sind, ohne teure »Number-Cruncher« benutzen zu müssen, und statt dessen auf die Rechenleistung zurückzugreifen, die vielerorts schon installiert ist und oft (beispielsweise nachts oder am Wochenende) brachliegt.

Anhang: Literaturverzeichnis

Computer-Animation ist ein komplexes Thema – zu komplex, als daß es im Rahmen dieses Handbuches zufriedenstellend erläutert werden könnte. Daher finden Sie im folgenden Abschnitt eine Literaturliste mit Empfehlungen, die dem interessierten Leser sicherlich weiterhelfen werden.

1) A Discussion on the Development of Motion Control Systems

   David Sturman

   Computer Graphics Lab

   New York Institute of Technology

   May 1986

2) Interactive Keyframe Animation of 3-D Articulated Models

   David Sturman

   Computer Graphics Lab

   New York Institute of Technology

3) Motion Simulation

   Richard Lundin

   Computer Graphics Lab

   New York Institute of Technology

   NicoGraph84

4) Applying Classical Techniques to Computer-Animation

   Glenn McQueen

   Computer Graphics Lab

   New York Institute of Technology

5) Leitfaden der Computer Grafik

   Bernd Willim

   Drei-R-Verlag Berlin, 1989

6) The RenderMan Companion

Steve Upstill

Addison-Wesley-Publishing Company

7) Occursus cum novo – Photorealistische Animation im Netzwerk

Wolfgang Leister, Heinrich Müller, Burkhard Neidecker, Achim Stößer

Abteilung Computer Grafik

Institut für Betriebs- und Dialogsysteme

Fakultät für Informatik

Universität Karlsruhe

Stichwortverzeichnis

A

Abbruch / Gelenke_fertig 67
Animations-Auftrag 92
Arbeitskörper 37
Ausrichten 36

B

Baum 13, 26
Baumstruktur 28
Beams 94
bewege_starr 68
Bilder 93
Bounding Box 58
Bruder 26

D

Darstellungsarten 33
Datensicherheit 98
Delete_Gelenk 67
Delta-Kompression 105
Diagnose-Datei 28
Doppelgänger 32
Double-Buffering 120
Drahtgittermodell 39
drehe_gelenk 68
drehe_starr 68
Drehpunkt 66

E

Eigenschaften 69
End-Zeit 93

F

fertig 74

G

Gelenke 14, 60
Gitterfunktion 33
Grafik-Programmiersprachen 131
Grafikkarte 108
Grund-Datei 93
Grundhaltung 29
Grundorientierung 29
Grundrichtung 29

H

Hardware-Aspekte 137
Hauptkörper 68
hierarchische Objektstruktur 27

I

int_good 71
int_quick 71
Interpolieren 44
Intervall 46

J

Job 92
Job-Datei 97

K

Kamera-Fahrten 74
Kamera-Regler 75
Kamerasequenz 50
Kamerastandort 75
Key-Frame 17
Keyframe-Systeme 131
Komplexität 135
Kompression 96
Kompressionsrate 108
Kontrollstrukuren 132
Koordinatensystem 13, 24
Körperbaum 76
Körperbeschreibungen 28
Körperstruktur 13
Kugelgelenke 61

L

Laden 97
Lichtfarbe 73
Lichtquelle 37
Lichtsequenz 50
Lichtsequenzen 72
Load_kam/li 74
Load_seq 74
Load_skl 74

M

Make_Anim 91
Manager 123
Modellierprogramme 28

O

Orientierungsrichtung 51

P

Pack 96
Partikel-Systeme 135
Pfad 17
Play 119
Plotfenster 18
Plotkörper-Liste 36
Preview-Menü 55
Probeberechnung 55

R

RAM-Animation 105
Raytrace-Animation 12
Reflections 11
Render 93
Richtungsflag 52, 75
Robotik 133
Roll-Winkel 75

S

Save_kam/li 74
Save_seq 74
Save_skl 74
Save_sz 74
Scan 94, 115
Script-Systeme 132
Sequenz 17, 41, 45
Sequenz-Datei 93
Sequenz-Fenster 45
Sequenz-Flags 75
Sequenz-Kombinationen 78
Sequenzfenster 18, 41
Show 95
show_gelenk 71
Simulation 133
Simulatoren 129
Skelett 14, 57
Skelett-Animation 17, 55
Skelettdatei 60
Skelettlinie 59
Skelettpunkt 59
Skelettpunkten 67
Snap 43, 45
Snap_Streifen 45
Sohn 26
Special-Menü 71
Speicherproblem 111
Speicherschutz 98
Speicherverbrauch 12, 106
Start-Zeit 93
Statusfenster 18
Strukturinformationen 28

T

Take 54
Take_Gelenke 67
Take_Snaps 70
Teleobjektiv 75
Toleranz 107

U

Ursprung 30

V

Vater 26
Vater-Sohn-Beziehung 26
Vektoren 24
Verbiegungen 83
Verdoppeln 32
verwandt 26
Verzerrungen 83
Videoband 105

W

Weltkoordinatensystem 24
Wire-Frame 17, 34
Wire-Frame-Animation 55

Z

Zeichentrickfilm 130
Zeitachse 42
Zeitcursor 41, 43, 45
Zeitintervall 56
Zeitskala 43, 45
Zielorientiert 134

Reflections Animator

Animation ist Faszination – die letzte im Bereich Computergrafik.
Durch schnelles Abspielen von Einzelbildern wird wie beim Film die Illusion einer fließenden Bewegung erzeugt.
Der Reflections-Animator ergänzt das Ray-Tracing-Programm Reflections desselben Autors zu einem phantastischen Animationssystem – und das zu einem phantastischen Preis!
Durch die ausgefeilten Programmtechniken gehören lange Wartezeiten der Vergangenheit an. Als Multitasking-fähiges Programm erledigt der Reflections-Animator seine Arbeit unbemerkt im Hintergrund, während Sie z.B. Texte schreiben.
Die U.S.S. Enterprise gleitet lautlos im All, plötzlich zünden die Motoren mit einem lauten Dröhnen, und das Raumschiff beschleunigt wie von einem Gummiband abgeschossen, um in den Tiefen des Universums zu verschwinden.
»Dynamische Objekte« ermöglichen Ihnen solche Effekte (Beschleunigen, Bremsen). So sind, nicht zuletzt durch die freien Kamerafahrten, eindrucksvolle Filme möglich.
Genießen Sie es, Künstler, Tricktechniker und Regisseur in einer Person zu sein!

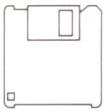
Hardware-Voraussetzung:

Alle Amiga mit mindestens 1 Mbyte RAM

Software-Voraussetzungen:

Kickstart 1.2, 1.3 oder OS 2.0
Reflections (ISBN 3-89090-272-X)

BOOK-WARE
Profi-Software zum Buchpreis

ISBN 3-89090-234-0

DM 98,– sFr 91,– öS 834,–
Unverbindliche Preisempfehlung